UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Pintar

IZGRADNJA PODATKOVNEGA SKLADIŠČA V DRUŽBI ZA TRGOVANJE Z ELEKTRIČNO ENERGIJO

DIPLOMSKO DELO

UNIVERZITETNI PROGRAM RAČUNALNIŠTVA IN INFORMATIKE
SMER INFORMATIKA

LJUBLJANA, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Pintar

IZGRADNJA PODATKOVNEGA SKLADIŠČA V DRUŽBI ZA TRGOVANJE Z ELEKTRIČNO ENERGIJO

DIPLOMSKO DELO

UNIVERZITETNI PROGRAM RAČUNALNIŠTVA IN INFORMATIKE
SMER INFORMATIKA

MENTOR: izr. prof. dr. Viljan Mahnič

LJUBLJANA, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Izgradnja podatkovnega skladišča v družbi za trgovanje z električno energijo

Tematika naloge:

V diplomskem delu predstavite projekt izgradnje podatkovnega skladišča v družbi za trgovanje z električno energijo. Za lažje razumevanje tematike najprej opišite postopke trgovanja in predstavite osnovne koncepte podatkovnih skladišč. Nato predstavite razloge za uvedbo podatkovnega skladišča v omenjeni družbi ter opišite njegovo arhitekturo in postopek njegovega modeliranja. Posebno pozornost posvetite implementaciji časovnih in datumskih dimenzij ter obravnavi velike količine podatkov v relacijskih in analitičnih podatkovnih bazah. Diplomsko delo zaključite z opisom odziva končnih uporabnikov in vpliva, ki ga ima podatkovno skladišče na poslovne procese družbe.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matej Pintar, z vpisno številko 63000360, sem avtor diplomskega dela z naslovom:

Izgradnja podatkovnega skladišča v družbi za trgovanje z električno energijo.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča,

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 8. februarja 2016 Podpis avtorja:

Zahvaljujem se mentorju Viljanu Mahniču za vso pomoč, nasvete, potrpežljivost in strokovno vodenje pri izdelavi diplomskega dela.

Zahvaljujem se Maruši, svoji družini, vsem prijateljem in sodelavcem. Kdor čaka, dočaka.

1 Kazalo

Povzetek

Namen diplomskega dela je predstaviti projekt »Izgradnja podatkovnega skladišča v družbi za trgovanje energije«.

Diplomsko delo je razdeljeno na tri sklope. V prvem sklopu so, z namenom boljšega razumevanja tematike v nadaljevanju diplomskega dela, na kratko predstavljene značilnosti trga z električno energijo na primeru slovenskega trga z električno energijo.

V drugem sklopu (tretje poglavje) so v ospredju osnove podatkovnega skladišča, razlike med dimenzijskim in normaliziranim podatkovnim skladiščem ter opisi vrst analitičnih podatkovnih baz.

V zadnjem sklopu je najprej na kratko opisano, kakšno je bilo stanje pred uvedbo podatkovnega skladišča v Skupini GEN-I in definicija njegove arhitekture. Nato so predstavljena orodja in procesi, s katerimi je bil načrtan dimenzijski in analitični model podatkovnega skladišča. V petem, šestem in sedmem poglavju sledi obravnava nekaterih najbolj zanimivih izzivov, ki so nastali pri implementaciji podatkovnega skladišča, kot so obravnavanje časovnih con, particioniranje tabel dejstev in indeksacije podatkovnega skladišča.

Diplomsko delo se zaključi s sklepno besedo o učinkih in vplivu vzpostavljenega podatkovnega skladišča v Skupini GEN-I.

KLJUČNE BESEDE:

- Podatkovno skladišče
- Dimenzijski podatkovni model
- MOLAP
- Particioniranje tabel
- ColumnStore indeks
- Trg z električno energijo
- Microsoft SQL Server

Abstract

The main objective of the thesis is to present a project »Building a data warehouse in an energy trading company«.

Thesis is divided into three sections. The first section contains a brief introduction to the characteristics of the electricity market in the case of Slovenian market.

Second section starts with theoretical background of data warehousing, its concepts of data retrieval, consolidation and periodicity. It also contains a description of differences between normalized and dimensional data warehouse and differences between various types of online analytical processing cubes.

The last section is divided into several chapters.

Fourth chapter contains a brief description of a situation regarding data fragmentation in company GEN-I before the data warehouse implementation, provides some basic information about the project, shows data warehouse architecture and provides definition of dimensional model. Fourth chapter is concluded with description of MOLAP database scheme.

Fifth, sixth and seventh chapters contains a description of some of the most exciting challenges and their solutions which were encountered during the development and implementation of the data warehouse.

Thesis is concluded with a final word about the impact the project had after introducing it to the end users.

KEYWORDS:

- Data warehouse
- Dimensional modeling
- MOLAP
- Table partitioning
- ColumnStore Index
- Electricity energy markets
- Microsoft SQL Server

1 Uvod

Na začetku 90. let dvajsetega stoletja je poslovni in računalniški svet pretresla revolucija – internet. Z njim so se pojavili novi poslovni modeli, ki so svojo konkurenčno prednost gradili na ponujanju neprestane dosegljivosti novih storitev (elektronska pošta, spletno nakupovanje, spletno bančništvo …) uporabnikom. Podjetja so, v želji izkoristiti trenutek, mrzlično razvijala specializirane internetne aplikacije, ki so bile navadno ločene aplikacije in slabo integrirane z obstoječimi sistemi. Število teh specializiranih internetnih aplikacij je z razvojem medmrežja naraščalo in podjetja so se začela srečevati s fragmentacijo in nekonsistentnostjo ključnih poslovnih podatkov, kar je oteževalo spremljanje poslovnih rezultatov in sprejemanje poslovnih odločitev.

Kot nalašč sta se ravno v tem obdobju objavili knjigi, ki sta reševali ta problem. Leta 1992 je Bill Inmon objavil knjigo »Building the Data Warehouse«, v kateri je definiral koncept centraliziranega grajenja sistema za poslovno odločanje, leta 1996 pa je Ralph Kimball objavil knjigo »Data Warehouse Toolkit«, v kateri je definiral koncept dimenzijskega grajenja sistema za poslovno odločanje. Obe knjigi sta prispevali k definiranju novega področja v računalništvu – grajenje podatkovnih skladišč oziroma sistemov za poslovno odločanje. Do danes so se po svetu zgradila številna podatkovna skladišča v različnih sektorjih (trgovina, bančništvo, zavarovalništvo …) – tudi v Sloveniji.

Na trgih električne energije v Evropi so se prav tako v začetku 90. let dvajsetega stoletja dogajale tektonske spremembe. Čeprav je bila internetna revolucija bolj na obrobju, pa sta liberalizacija in uvajanje tržnih mehanizmov trg dodobra pretresla. Novonastala konkurenca je prej monopolistična podjetja prisilila v iskanje novih tržnih priložnosti, ustvarjanju novih, bolj dinamičnih produktov in v trgovanje na borzah. Obvladovanje tveganj, povezanih s trgovanjem na trgih z električno energijo, je tako postajalo vedno bolj pomembno. Posledično je količina podatkov eksplodirala, eksplodiralo je število aplikacij in mnogo podjetij je moralo za ustrezno obvladovanje tveganj in konsolidacijo poseči po podatkovnih skladiščih.

V Slovenijo je revolucija na trgih z električno energijo prišla z nekajletno zamudo, kar se tudi pozna po številu podatkovnih skladišč, ki bi celovito pokrivala tveganja pri trgovanju in prodaji električne energije. Trenutno v slovenski energetiki, razen v Skupini GEN-I,[1] ni podatkovnega skladišča te vrste, še posebej pa ne take velikosti.

Ker je v slovenskem prostoru o podatkovnih skladiščih te vrste dostopno relativno malo informacij, smo se odločili, da v tem diplomskem delu predstavim projekt izgradnje podatkovnega skladišča v Skupini GEN-I. Ker je podatkovno skladišče preobsežna tema za eno diplomsko nalogo, smo se odločili, da se predstavi samo en delček, in sicer področje vodenja portfelja pogodb pri Trgovanju in Prodaji.

[1] Lastna raziskava.

Pogledali si bomo, kako je definiran trg z električno energijo, ter teoretično predstavili področje podatkovnih skladišč. Predstavili bomo tudi orodja in postopke, s pomočjo katerih smo modelirali dimenzijski model, in preverili, kako smo reševali problematiko zapisa časovne značke poslovnih dogodkov.

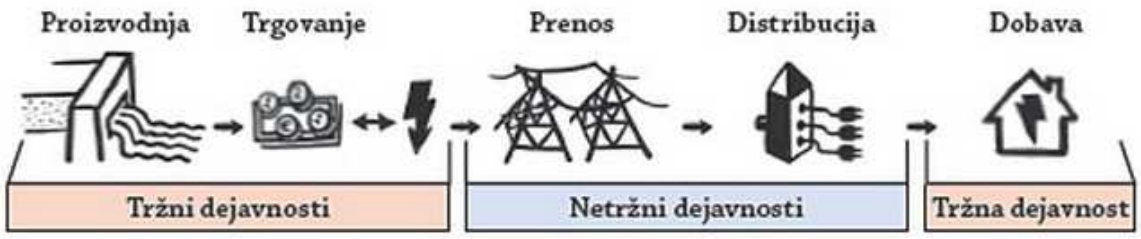
Diplomsko delo bomo zaključili z obravnavo nekaterih problemov, na katere naletimo pri obdelavi velike količine podatkov.

2 Trg z električno energijo

Električna energija je ena izmed najbolj uporabnih oblik energije, saj z njo napajamo električne naprave, brez katerih si danes težko predstavljamo sodobno življenje. Električne naprave tako najdemo v industriji, prometu, doma in na še številnih drugih področjih. Slaba stran električne energije je, da je neposredno ne moremo skladiščiti,[2] saj gre za prehodno obliko energije – delo. Zaradi tega so elektroenergetski sistemi po državah zasnovani tako, da proizvodnjo električne energije prilagajajo povpraševanju.

V preteklosti so države oskrbo z električno energijo organizirale kot javno storitev, ki je bila izvzeta iz delovanja trga. Segmenti oskrbe (proizvodnja, prenos, distribucija in končna dobava) so bili vertikalno integrirani v gospodarske subjekte (večinoma v državni lasti), ki so de facto imeli monopol nad dobavo do gospodinjstev in poslovnih odjemalcev znotraj nekega določenega geografskega območja. Zato so države to dejavnost strogo regulirale in določale končno ceno električne energije [10].

Leta 1989 so začeli v Veliki Britaniji, po zgledu liberalizacije v drugih gospodarskih sektorjih, s procesom uvajanja tržnih mehanizmov v elektroenergetski sistem. Postavili so nov model energetskega trga, po katerem so vertikalno ločili potencialno tržne segmente oskrbe (proizvodnja in končna dobava) od močno reguliranih segmentov oskrbe (distribucija in prenos električne energije). V segmentih proizvodnje in končne dobave so nato izvedli še horizontalno ločitev z namenom zagotovitve zadostnega števila med seboj konkurenčnih subjektov na trgu, v segmentih distribucije in prenosa električne energije pa horizontalno integracijo, ko so združili infrastrukturo pod »eno« streho – sistemskim operaterjem (angl. *Transmission System Operator – TSO*) [10].



Slika 1: Shematski prikaz udeležencev na trgu z električno energijo [1]

[2] Električno energijo lahko na primer skladiščimo v obliki kemične energije (akumulatorji), v obliki mehanske kinetične energije (vztrajniki), v obliki potencialne energije (črpalne in hidro elektrarne), toplote (talinske soli) in s shranjevanjem »naboja« (superprevodniki). Pri skladiščenju električne energije ni toliko problem tehnologije, kot je problem z gospodarnostjo le-tega (izgube pri pretvorbi, stroški, neučinkovitost ...).

Reforma trga se je izkazala za zelo zahtevno nalogo, vendar pa so bili rezultati reforme dovolj dobri [7], da je Evropska komisija z direktivo 96/92/EG zavezala članice Evropske unije vzpostaviti skupen notranji trg z električno energijo po britanskem zgledu. V naslednjih letih so se tako vzpostavili trgi na Finskem (1997), v Nemčiji (1999), na Švedskem (1998), v Avstriji (2001) in drugih državah [8].

V Sloveniji je bil leta 1999 sprejet Energetski zakon in uredba o načinu izvajanja gospodarskih javnih služb za prenos in distribucijo električne energije, ki je pomenil začetek vzpostavljanja trga z električno energijo. Notranji trg se je za slovenske proizvajalce odprl aprila 2001, ko so upravičeni odjemalci že lahko izbirali svojega dobavitelja električne energije, od 1. julija 2004 dalje so lahko svojega dobavitelja izbirali poslovni odjemalci, od 1. julija 2007 pa se je trg odprl za vse odjemalce električne energije [9].

Poglejmo si torej, kako so ti novoustanovljeni trgi sestavljeni.

2.1 Organizacijska struktura trga z električno energijo

Trgi električne energije v državah EU se med seboj strukturno ne razlikujejo veliko[3] in načeloma sledijo priporočilom direktive Evropske komisije. Za razumevanje trgov električne energije tako zadostuje pregled organizacijske strukture slovenskega elektroenergetskega sistema:

a) **Proizvajalec**: V vlogi proizvajalca lahko nastopata tako proizvajalec električne energije (npr.: hidroelektrarna) kot trgovec z električno energijo. Oba imata možnost prodaje električne energije na organiziranem trgu z električno energije neposredno dobavitelju in/ali neposredno odjemalcu.

b) **Odjemalec**: Odjemalec je pravna ali fizična oseba, ki ji proizvajalec ali dobavitelj proda električno energijo. Odjemalce lahko delimo na več skupin, in sicer glede na velikost odjema električne energije [2]:

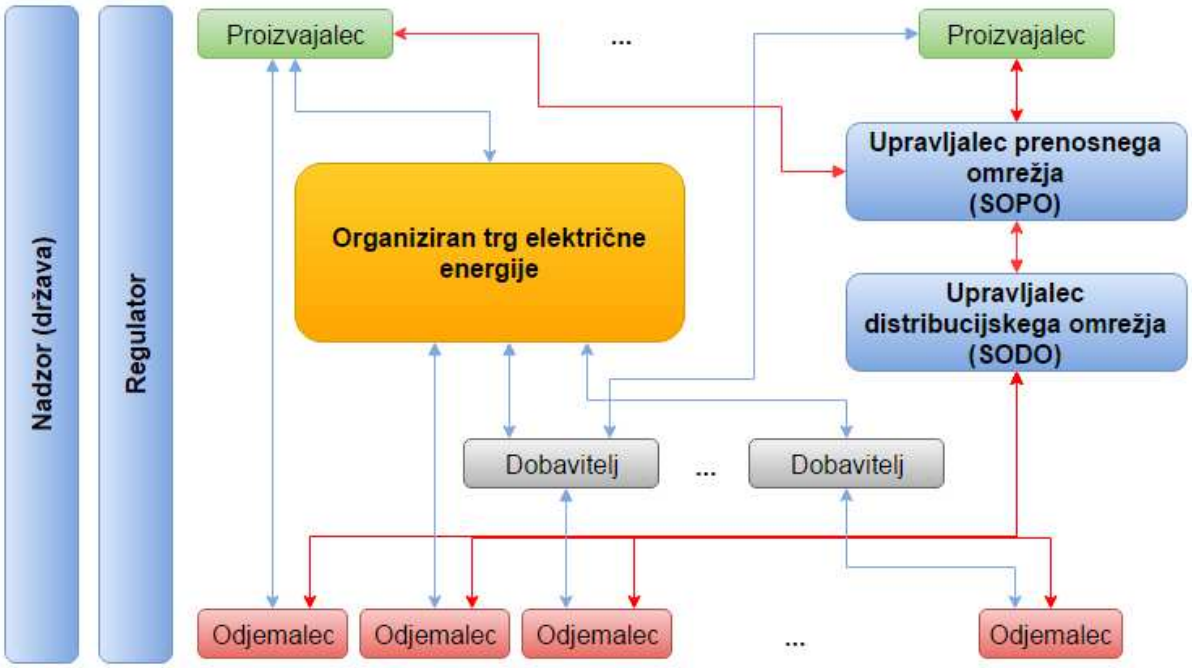
   i. Industrijske konglomerate, železnice (območje TWh na leto).
   ii. Velike industrijske odjemalce (nekaj GWh na leto).
   iii. Srednje velike industrijske odjemalce (pod GWh na leto).
   iv. Male poslovne odjemalce in gospodinjstva (območje nekaj MWh na leto).

   Odjemalci v skupinah i.–iii. imajo v veliki večini s proizvajalci ali trgovci vzpostavljen lasten portfelj dobave električne energije, ki ga aktivno upravljajo glede na dogajanje na trgih električne energije. Z odjemalci v skupini iv. dobavitelji ali trgovci sklepajo odprte pogodbe po fiksni ceni.

[3] Za strukturo električnih trgov v drugih državah glej [6].

c) **Dobavitelj**: Dobavitelji električne energije so lahko trgovci, zastopniki, posredniki ali proizvajalci električne energije, ki imajo ustrezno licenco za trgovanje, zastopanje oziroma posredovanje in proizvodnjo električne energije.
d) **Organizator trga električne energije**: Organizator trga v Sloveniji je družba Borzen, ki izvaja naloge vodenja bilančne sheme, evidentiranja bilateralnih pogodb, izdelavo okvirne obratovalne napovedi (vozni red) ter bilančni obračun in finančno poravnavo poslov, povezanih s predhodno navedenimi nalogami [11].
e) **Upravljavec prenosnega omrežja (SOPO)**: Upravljavec prenosnega omrežja skrbi za prenos električne energije za potrebe slovenskega in tudi mednarodnega trga z električno energijo. SOPO upravlja čezmejne prenosne zmogljivosti.
f) **Upravljavec distribucijskega omrežja (SODO)**: Upravljavec distribucijskega omrežja izvaja gospodarsko javno službo distribucijskega operaterja električne energije – dobavo električne energije do končnega odjemalca. Skrbi tudi za načrtovanje razvoja distribucijskega omrežja, njegovo izgradnjo, vodenje in obratovanje.
g) **Regulator (navadno v državni lasti)**: Regulatorji električnega trga imajo v državah EU sicer različen obseg nalog in različne naloge, vendar pa imajo vsi enaka temeljna izhodišča. Ta so zagotavljanje nepristranskosti in učinkovite konkurence na trgu električne energije ter zanesljive in nemotene oskrbe odjemalcev električne energije.
h) **Borza oziroma organiziran trg električne energije**: Borza proizvajalcem, dobaviteljem in odjemalcem ponuja enostavno, transparentno, nepristransko in varno trgovanje s standardnimi produkti električne energije. Borza zagotavlja pregledno in nediskriminatorno oblikovanje sklenitvene cene.

Slika 2 prikazuje povezave med posameznimi udeleženci trga z električno energijo. Rdeče povezave prikazujejo fizično omrežje, medtem ko so modre povezave pogodbene narave. Nad celotnim dogajanjem bdi regulator trga (država).



Slika 2: Organizacijska struktura trga z električno energijo v Sloveniji [2]

Slika 2 nam tudi prikazuje naslednje povezave med udeleženci na trgu z električno energijo:

- Obstajajo odjemalci, ki imajo pogodbo o odjemu električne energije sklenjeno neposredno s proizvajalcem – dvostransko trgovanje.
- Obstajajo dobavitelji, ki kupujejo električno energijo na organiziranem trgu električne energije in jo nato dobavljajo odjemalcem – trgovanje na borzi.
- Obstajajo odjemalci, ki kupujejo električno energijo na organiziranem trgu električne energije.
- Obstajajo proizvajalci, ki prodajajo električno energijo na organiziranem trgu električne energije.

2.2 Trgi z električno energijo

2.2.1 Terminski trg

Terminski trg z električno energijo je trg, na katerem se trguje z električno energijo za časovna obdobja od enega dneva do več let vnaprej. Terminski trg ponuja dve vrsti trgovanja:

a) **Dvostransko**,[4] pri katerem udeleženca skleneta nestandardno terminsko pogodbo (angl. *forward*) o prodaji ali nakupu električne energije z rokom izpolnitve na določen dan v prihodnosti, po ceni, dogovorjeni na dan sklenitve pogodbe [4]. Pri tem je pogodba glede pogodbenih določil (produkt, način plačila, plačilni roki, garancije ...) prilagojena potrebam obeh udeležencev.
b) **Trgovanje na borzi**, pri katerem udeleženca skleneta standardno terminsko pogodbo (angl. *futures*). Standardne terminske pogodbe so terminske pogodbe, ki imajo standardizirane produkte[5] in standardizirana pogodbena določila. Na borzi se trguje le s produkti pasovne ali trapezne električne energije.

Večina trgovanja na terminskem trgu se odvije na dvostranskem trgu, saj le-ta ponuja večjo fleksibilnost velikim odjemalcem, ki si tako od proizvajalcev zagotovijo odjem električne energije za daljše obdobje glede na njihovo napoved porabe v prihodnosti. Ker pa se z nestandardnimi terminskimi pogodbami ne trguje na borzi, obstaja večje tveganje za neizpolnitev pogodbenih določil ene izmed strani. Po drugi strani pa trgovanje na borzi prinaša manjše tveganje, saj se standardiziran terminski produkt dnevno vrednoti in se izračunava gibljivo kritje (angl. *variation margin*).[6]

Povezava med trgovanjem na dvostranskem trgu in trgovanjem na borzi je tako imenovano varovanje tveganja (angl. *hedging*) fizične pozicije,[7] s katerim se prodajalec ali kupec zaščiti pred nihanjem cen na trgu električne energije.

2.2.2 Promptni trg

Promptni trg (ali SPOT) je trg, na katerem se trguje z električno energijo za dan vnaprej (angl. *day ahead*) in znotraj dneva. Na trgih, kjer je organiziran trg (borza) dobro razvit in likviden,

[4] Gre za *Over The Counter* (OTC) trg.

[5] Glej poglavje »Standardni produkti na trgu električne energije«.

[6] *Variation margin* je razlika med dogovorjeno ceno standardne terminske pogodbe in dejansko ceno te pogodbe na borzi. V odvisnosti od pozicije – nakupna (ali prodajna) se v primeru višanja cene na borzi prejema (vplačuje), v primeru nižanja cene na borzi pa vplačuje (prejema) denar.

[7] Trgovec sklene z odjemalcem dolgoročno prodajno *Futures* pogodbo. Da se trgovec vsaj malo zaščiti pred nihanjem cen na trgu z električno energijo, hkrati sklene na borzi nakupno *Forward* pogodbo s standardnim produktom. Če bi imeli *Futures* in *Forward* pogodba enak standardiziran produkt, ceno in bi obe sklenili na istem trgu, bi se ti dve pogodbi med seboj profitno izničili.

se trguje večinoma prek borze, medtem ko se na manj razvitih in likvidnih trgih trguje dvostransko.

2.2.3 Izravnalni trg

Električno omrežje mora biti vedno poravnano. To pomeni, da mora biti vsota med proizvedeno in porabljeno električno energijo enaka nič. Izravnalni trg je namenjen ravno izravnavanju med obratovalno napovedjo udeležencev in dejansko proizvedeno in porabljeno električno energije. Vsak udeleženec mora posredovati upravljavcu prenosnega omrežja (SOPO) obratovalno napoved pritokov in odtokov električne energije, ki jih bodo imeli znotraj države in čez meje.

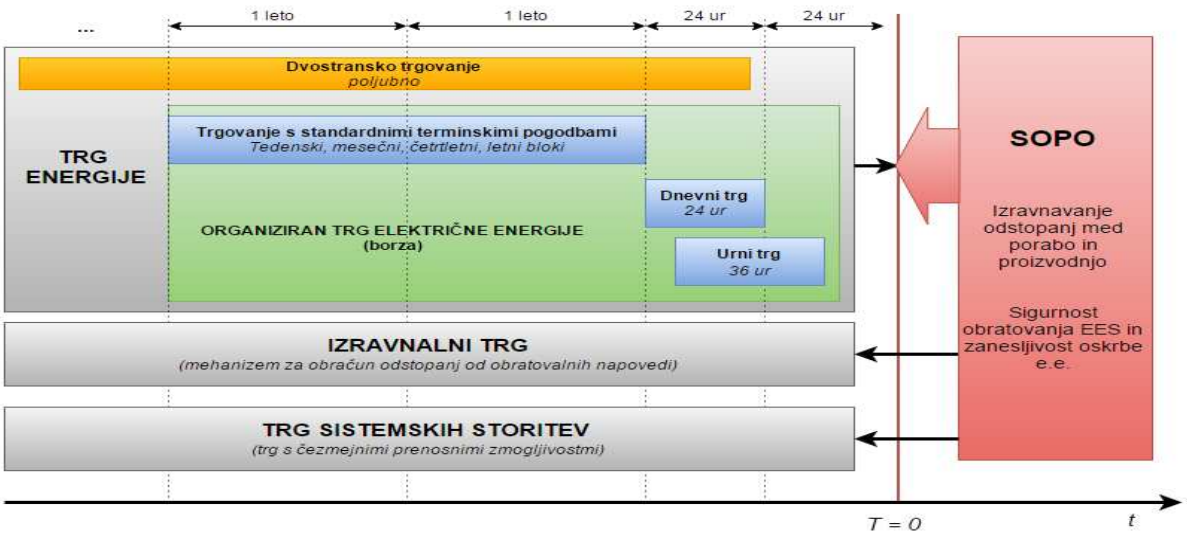
Velikokrat se zgodi, da se realna proizvodnja in poraba električne energije razlikujeta od obratovalne napovedi, zato mora SOPO v primeru pomanjkanja ali presežka v omrežju znotraj države ustrezno povečati ali zmanjšati količino električne energije. To lahko stori s povečanjem ali zmanjšanjem proizvodnje znotraj države ali pa z nakupom ali prodajo električne energije v tujini. Nastali stroški zaradi izravnave SOPO bremenijo tiste udeležence, ki v tistem trenutku niso bili izravnani [12].

2.2.4 Trg s čezmejnimi prenosnimi zmogljivostmi

»Čezmejna prenosna zmogljivost je razpoložljivost interkonekcijske povezave med dvema sosednjima elektroenergetskima sistemoma, ki je namenjena komercialni uporabi. Količino čezmejnih prenosnih zmogljivosti določi sistemski operater (TSO) v sodelovanju s sistemskim operaterjem sosednje države in ob upoštevanju meril zagotavljanja zanesljivosti obratovanja prenosnega omrežja« [12].

Upravljavci prenosnega omrežja (SOPO) upravljajo tudi čezmejne prenosne zmogljivosti, ki jih prodajajo na letni, mesečni in dnevni ravni glede na zmožnost prenosa daljnovodov iz ene države v drugo.

Slika 3 prikazuje trge električne energije in njihovo razmerje do časa dobave (T = 0). Trgovanje na promptnem (SPOT) trgu se začne dva dneva pred dobavo in zaključi tik pred samo dobavo električne energije. Upravljavec prenosnega omrežja nato izmeri razliko med obratovalno napovedjo in dejanskim stanjem omrežja ter obračuna odstopanja, pri tem pa skrbi za zanesljivost obratovanja elektroenergetskega sistema.



Slika 3: Trgi električne energije [2]

2.3 Regulacijsko območje in bilančne skupine

Vsak trg z električno energijo je razdeljen na eno ali več regulacijskih območij. Če želi proizvajalec, dobavitelj ali odjemalec s svojimi oddajno-odjemnimi mesti sodelovati na trgu z električno energijo, mora z organizatorjem trga električne energije najprej skleniti bilančno pogodbo. Z njo proizvajalec, dobavitelj ali odjemalec uredi dobavo izravnalne energije in finančno poravnavo v primeru energetske neporavnanosti, s čimer se uvrsti v regulacijsko območje kot odgovorni bilančne skupine in pridobi status člana regulacijskega območja. Člani regulacijskega območja se lahko združujejo/razdružujejo v bilančne skupine, vrh katerih predstavlja odgovorni bilančne skupine, pod katerim se lahko zvrsti poljubno mnogo hierarhično nižjih članov bilančne skupine oziroma podskupin. Namen ustanovitve bilančne skupine je lažje obvladovanje in upravljanje tveganj odstopanj odgovornega bilančne skupine in hierarhično nižjih članov bilančne skupine [14].

Slika 4 prikazuje sestavo regulacijskega območja in bilančne skupine. Neko oddajno/odjemno mesto je lahko član samo ene bilančne skupine.

Slika 4: Sestava regulacijskega območja [2]

2.4 Vrste pogodb na trgu električne energije

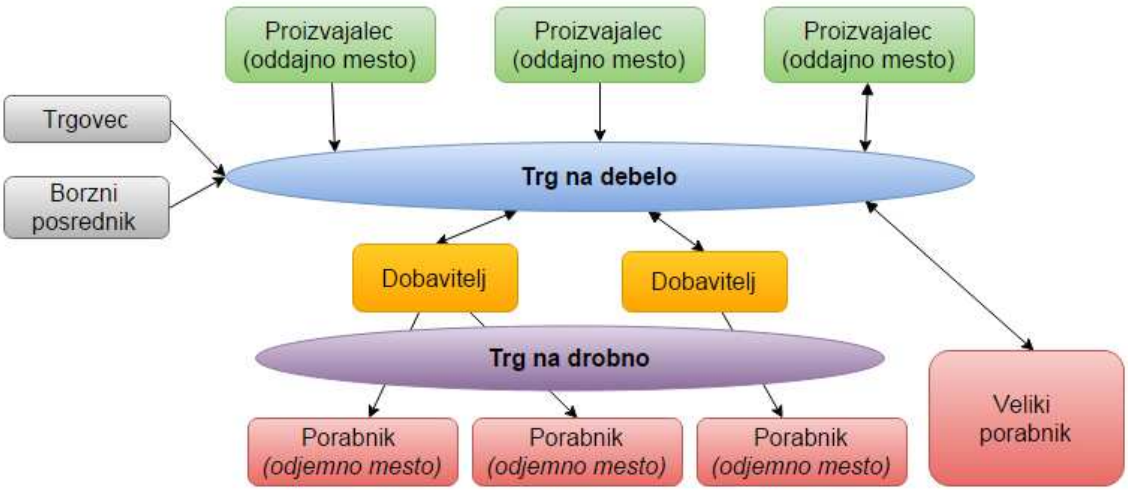
Pri trgovanju na trgu električne energije poznamo [14]:

a) **Zaprto pogodbo**: gre za pravni posel in druga razmerja med člani regulacijskega območja organiziranega trga z električno energijo, ki vsebuje količino dobavljene električne energije v obračunskem intervalu.[8] Pri obračunu odstopanj se količine, zapisane v pogodbi, upoštevajo nespremenjene, neodvisno od realizacije. Med zaprte pogodbe sodijo tudi pogodbe z uporabo čezmejnih prenosnih zmogljivosti, ki se štejejo kot zaprte pogodbe za dobavo električne energije prek meja in ki vključujejo tudi uporabo pravic čezmejnega prenosa. Zaprte pogodbe se lahko sklepajo med vsemi udeleženci na trgu električne energije, navadno pa se sklepajo na trgu na debelo.

b) **Odprto pogodbo**: gre za pravni posel in druga razmerja, ki določajo bilančno pripadnost oddajno-odjemnih mest. Odprta pogodba vsebuje predvideno obratovalno napoved po obračunskih intervalih. Odprte pogodbe se navadno sklepajo na trgu na drobno.

Če na trg z električno energijo pogledamo skozi vidik vrste pogodbe, potem se nam ta razdeli na trg na debelo in na trg na drobno. Na trgu na debelo trgujemo z električno energijo na borzi ali dvostransko z ostalimi trgovci, velikimi odjemalci in proizvajalci, medtem ko na trgu na drobno električno energijo prodajamo malim končnim odjemalcem oziroma malim proizvajalcem.

[8] Obračunski interval je časovno obdobje, v katerem se ugotavljajo odstopanja. Velika večina trgov ima obračunski interval eno (1) uro, v Nemčiji in v Avstriji pa je le-ta 15-minuten.



Slika 5: Električni trg na debelo in drobno [2]

2.5 Standardni produkti na trgu električne energije

Kot smo navedli v poglavju 2.2.1, borze ponujajo standardizirane produkte električne energije. Standardiziran produkt je definiran z urnim blokom ter obdobjem dobave.

Poznamo naslednje urne bloke energije:

- **Pasovna energija** (angl. *base load*): električna energija v bloku od 00.00 do 24.00 za dneve ponedeljek–nedelja.
- **Trapezna energija** (angl. *peak load*):[9] električna energija v bloku od 08.00 do 20.00 za dneve ponedeljek–petek, brez praznikov.
- **Nočna energija** (angl. *off-peak load*): električna energija v bloku:
  - ponedeljek–petek, brez praznikov, znotraj dneva od 00.00 do 08.00 in od 20.00 do 24.00,
  - sobota–nedelja in prazniki: od 00.00 do 24.00.
- **Urna energija:** trgovanje s 24 urami enega dneva.

Na borzah se na terminskem trgu trguje s pasovno energijo in trapezno energijo z obdobjem dobave dan, vikend, teden, mesec, četrtletje in leto, medtem ko se na promptnem trgu trguje z vsemi vrstami urnih blokov energije. Nekatere borze omogočajo tudi trgovanje z nestandardnimi bloki energije, s katerimi udeleženci na borzi optimizirajo trgovanje.

[9] Gre za t. i. evropsko trapezno energijo. V drugih državah je lahko trapezna energija definirana drugače, v Sloveniji npr. kot blok od 06.00 do 22.00 za dneve ponedeljek–petek, brez praznikov.

3 Kaj je podatkovno skladišče?

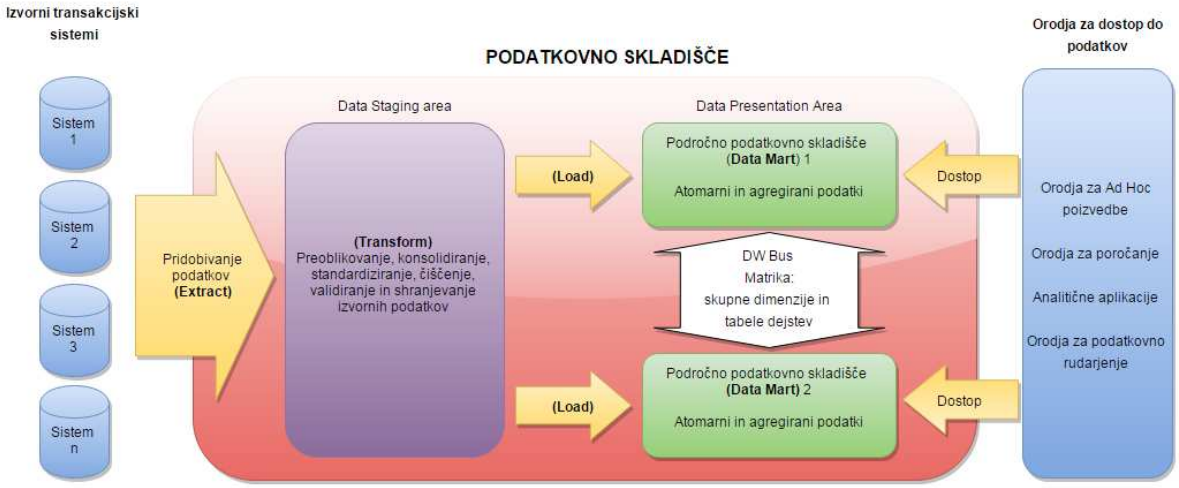
Bill Inmon je podatkovno skladišče definiral kot: »Podatkovno skladišče je zbirka entitetno usmerjenih, integriranih, časovno odvisnih in nespreminjajočih se podatkov, ki so namenjeni podpori odločitvenim procesom« [16].

Ralph Kimbal je postregel s podobno definicijo, in sicer: »Podatkovno skladišče je sistem, ki pridobiva, čisti in usklajuje podatke iz izvornih sistemov, ter jih shranjuje v dimenzijsko podatkovno shrambo (angl. *dimensional data store*) za namen poizvedb in analiz, ki so namenjene podpori odločitvenim procesom« [3].

Obe definiciji govorita o sistemu, ki integrira podatke iz različnih izvornih sistemov, z namenom podpore odločitvenih sistemov, razlikujeta pa se pri načinu fizične implementacije podatkovne sheme. Po Inmonu je podatkovno skladišče fizično implementirano kot normalizirano podatkovno skladišče, medtem ko je po Kimballu podatkovno skladišče fizično implementirano kot dimenzijsko podatkovno skladišče.

Po mojem mnenju pa podatkovno skladišče še najbolje definira Vincent Rainaldi, ki združuje obe zgoraj omenjeni definiciji v eno:
»Podatkovno skladišče je sistem, ki pridobiva in združuje podatke iz izvornih transakcijskih sistemov v dimenzijsko ali normalizirano podatkovno skladišče. Navadno vsebuje večletno zgodovino in se uporablja za poizvedbe poslovne inteligence ali drugih analitičnih postopkov. Pridobivanje podatkov poteka periodično v paketu, in ne vsakokrat, ko se spremenijo podatki v izvornih sistemih« [15].

V nadaljevanju bomo nekatere termine iz te definicije bolj podrobno opisali in razložili, pri tem pa si bomo pomagali s sliko 6, ki prikazuje osnovne elemente podatkovnega skladišča.



Slika 6: Osnovni elementi podatkovnega skladišča [3]

3.1 Pridobivanje podatkov

Pridobivanje podatkov podatkovnega skladišča poteka prek množice opravil in postopkov, ki so združeni v t. i. ETL-sistem. ETL-sistem je okrajšava za pridobivanje podatkov iz izvornih sistemov (**E**xtract), preoblikovanje le-teh v ustrezno obliko (**T**ransform) in njihovo shranjevanje v ustrezno podatkovno shemo ciljnega sistema (**L**oad). ETL-sistemi se ne uporabljajo samo za polnjenje podatkovnega skladišča, ampak se pogostokrat uporabljajo pri kakršnem koli premikanju podatkov iz enega v drug sistem.

V postopku preoblikovanja podatkov lahko podatke spreminjamo, da ustrezajo pogojem in formatu podatkovnega skladišča (npr.: uskladimo podatkovne tipe), lahko dodajamo nove podatke, ki so izpeljani iz obstoječih podatkov, ali pa pridobljene podatke uporabimo za validacijo podatkov v izvornih sistemih.

Večina ETL-sistemov vsebuje tudi mehanizme za čiščenje podatkov, preden so le-ti shranjeni v ciljni sistem. Čiščenje podatkov je proces identificiranja in popravljanja nepravilnih podatkov s pomočjo pravil, ki določijo, kdaj je neki podatek nepravilen. Če pravilo odloči, da je neki podatek pravilen, se le-ta shrani v podatkovno skladišče. V primeru nepravilnega podatka pa imamo na voljo tri možnosti in v odvisnosti od situacije podatek zavrnemo, ga popravimo s pomočjo pravil zagotavljanja kakovosti podatkov ali pa ga nespremenjenega shranimo v podatkovno skladišče.

Vse zgoraj opisane operacije potekajo na posebnem področju podatkovnega skladišča – področju za preoblikovanje podatkov (angl. *data staging area*).

3.2 Združevanje podatkov

Organizacije imajo lahko več transakcijskih sistemov, ki podpirajo različne poslovne procese. Na primer skupina GEN-I ima več aplikacij, ki pokrivajo trgovanje z električno energijo, prodajo električne energije velikim odjemalcem[10] v Sloveniji, prodajo električne energije malim poslovnim odjemalcem v Sloveniji, prodajo električne energije v tujini, aplikacijo za stroške, ki nastanejo pri trgovanju z električno energijo, in druge.

Podatkovno skladišče s pomočjo ETL-sistema združuje vse te aplikacije v enotno sliko glede na:

- **različno razpoložljivost podatkov**: nekateri podatki so na voljo samo v sistemu A, niso pa na voljo v sistemu B,
- **različne časovne značke**: poslovni dogodki so v sistemu A zapisani v urah, v sistemu B so agregirani na mesečnem nivoju,
- **različne definicije:** mera Vrednost pogodbe v sistemu A vsebuje davek, v sistemu B pa ne,
- **konverzije**: mera Vrednost pogodbe je v sistemu A samo v valuti EUR, v sistemu B pa lahko vsebuje tudi ostale valute,
- **združevanje podatkov iz različnih sistemov po nekem ključu, da preprečimo podvojene/potrojene ali nesmiselne primerjave**: partner X ima v sistemu A drugačno identifikacijo kot v sistemu B, v podatkovnem skladišču ga vodimo kot enega partnerja, in ne dveh.

Združevanje podatkov prav tako poteka na področju za preoblikovanje podatkov podatkovnega skladišča.

3.3 Periodičnost

Pridobivanje in združevanje podatkov ni enkraten dogodek, ampak gre za proces, ki se izvaja v ponavljajočih se intervalih, navadno enkrat ali večkrat dnevno. Če bi šlo za enkraten proces, bi podatki kmalu postali zastareli in čez nekaj časa neuporabni. Periodo osveževanja (pridobivanje in združevanja podatkov) določamo glede na poslovne zahteve in frekvenco spreminjanja podatkov v izvornih sistemih. Če se podatki v izvornih sistemih spremenijo enkrat tedensko, ni smiselno postaviti dnevne periode osveževanja, če pa se podatki neprestano spreminjajo in so spremembe bistvenega pomena za poslovni proces, je smiselno določiti večkratno osveževanje znotraj dneva.

[10] Glej poglavje 2.1

3.4 Dimenzijsko podatkovno skladišče

Ko smo podatke s pomočjo ETL-sistema ustrezno obdelali, je čas, da jih shranimo v ustrezno področno podatkovno skladišče oziroma v normalizirano podatkovno skladišče ter jih prek različnih orodij za dostop do podatkov ponudimo (predstavimo) uporabnikom.

Dimenzijsko podatkovno skladišče *(*angl. *Dimension Data Store)* je ena ali več podatkovnih baz, ki vsebujejo množico področnih podatkovnih skladišč. Značilnost dimenzijskega podatkovnega skladišča je njegova denormalizacija relacijske podatkovne baze. Denormalizirana relacijska podatkovna baza je tista baza, katere tabele vsebujejo redundantne in soodvisne atribute.[11]

Področno podatkovno skladišče (angl. *Data mart*) je skupek povezanih tabel dejstev ter njim pripadajočih dimenzij, s katerimi opisujemo in merimo dogodke nekega določenega področja.

Tabela dejstev (angl. *Fact Table*) je tabela, ki vsebuje samo ključe pripadajočih dimenzij ter mer določenega področja.

Dimenzijska tabela (angl. *Dimension*) je tabela, ki vsebuje opis, kategorije in lastnosti določenega področja.

Atribut (angl. *Attribute*) je stolpec v dimenziji ali tabeli dejstev. Stolpec v tabeli dejstev, ki ni ključ dimenzij, se imenuje mera (angl. *Measure*).

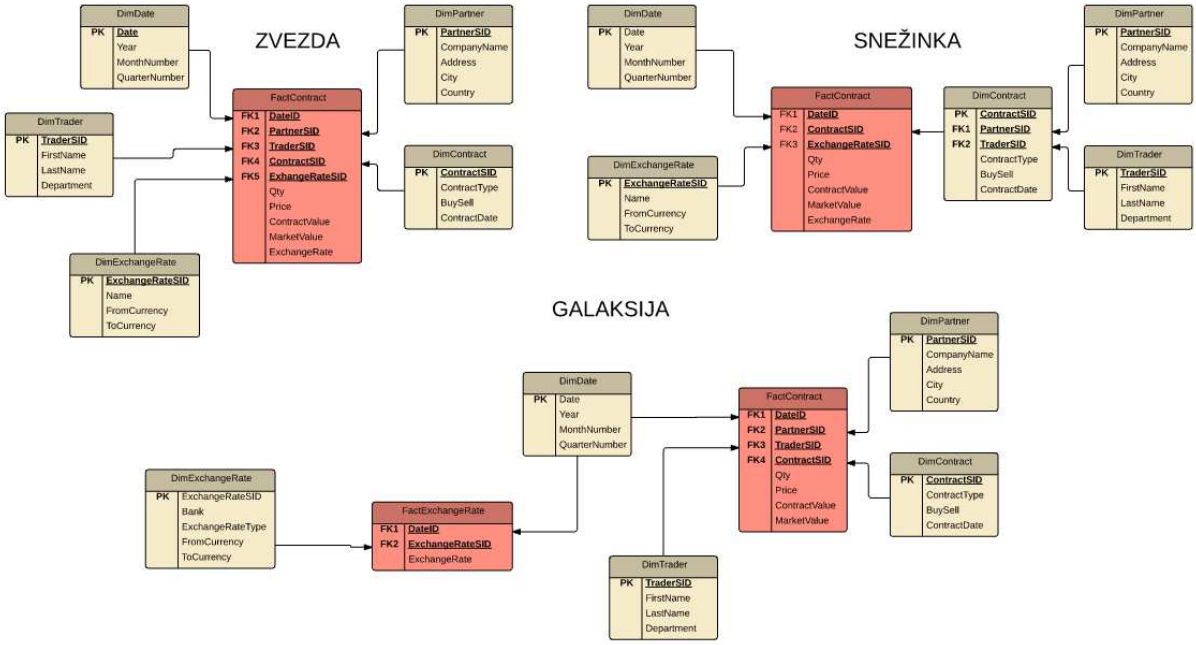
Kot je zapisal Vincent Rainardi [15], lahko dimenzijsko podatkovno skladišče fizično realiziramo v treh oblikah: kot zvezdo (angl. *Star*), snežinko (angl. *Snowflake*) ali galaksijo (angl. *Galaxy/Constellation*).

Kot vidimo na sliki 7, kjer so prikazane vse tri vrste, imamo zvezdno shemo takrat, ko nobena izmed dimenzij nima poddimenzij (oziroma podtabel), snežinkasto shemo dobimo, ko ima katera izmed dimenzij poddimenzije. Poddimenzijo si lahko predstavljamo kot dimenzijo, katere ključ je vsebovan v drugi dimenziji.[12]

---

[11] Za dodatno razlago denormalizacije glej [20].

[12] Primer poddimenzije je entiteta »sestavni del« neke naprave. Kot naddimenzija te dimenzije bi tako nastopala entiteta »naprava«, ki vsebuje več sestavnih delov entitete »sestavni del«.



Slika 7: Vrste shem dimenzijskih podatkovnih skladišč

Značilnost galaktične sheme je v tem, da imamo več povezanih tabel dejstev, ki so obkrožene s skupnimi dimenzijami.

Prednost zvezdne sheme v primerjavi s snežinkastimi in galaktičnimi shemami je v njeni enostavni shemi in v lažji implementaciji ETL-procesa. Prednost snežinkaste sheme je v manjši redundanci podatkov, kar pomeni manjšo porabo diskovnega prostora v primerjavi z zvezdno shemo. V preteklosti je bila prednost snežinkaste sheme tudi v boljšem delovanju nekaterih analitičnih orodij, vendar pa so danes na trgu na voljo orodja, ki so optimizirana na zvezdno shemo, tako da je danes ta prednost vprašljiva. Prednost galaktične sheme v primerjavi z drugima dvema je v boljšem opisu poslovnih dogodkov nekega področja s pomočjo več tabel dejstev [15].

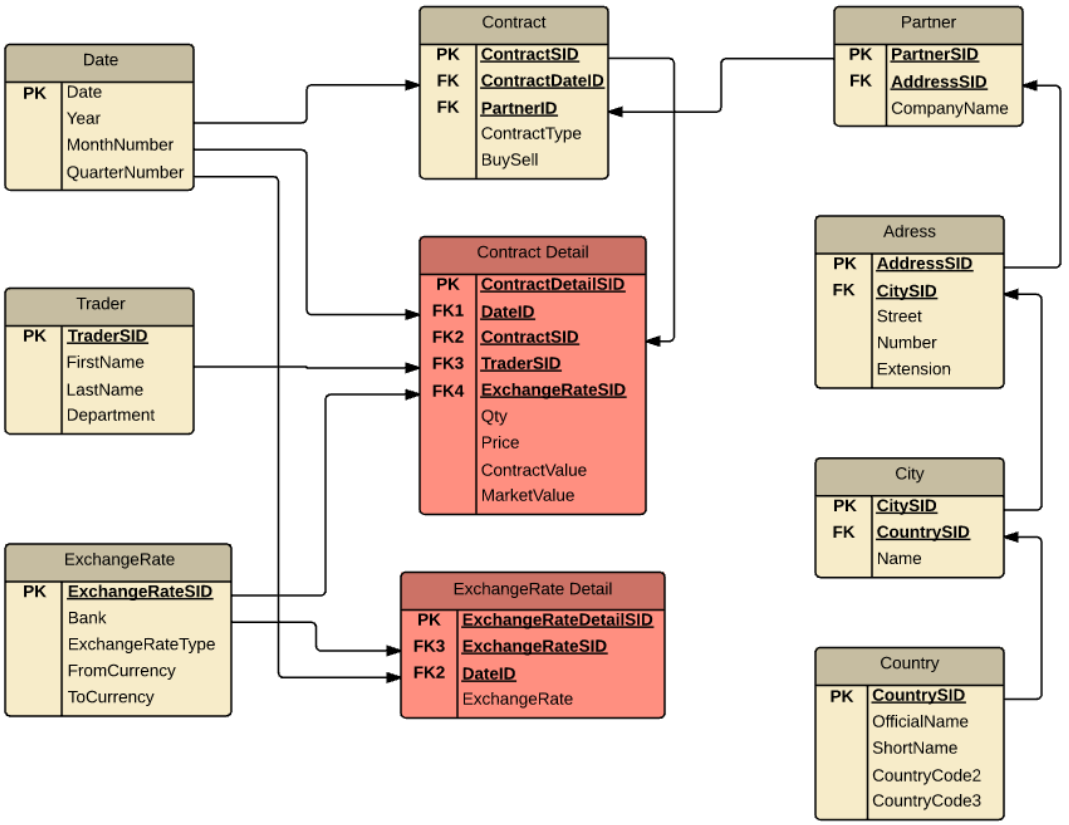
3.5 Normalizirano podatkovno skladišče

Drugi tip podatkovnih skladišč, poleg dimenzijskega, je normalizirano podatkovno skladišče, ki podatke shranjuje v relacijske podatkovne baze, pri katerih ne prihaja do redundantnosti in soodvisnosti podatkov, oziroma je stopnja redundance in soodvisnosti podatkov zelo nizka. Relacijska podatkovna baza vsebuje entitete tabele, ki so med seboj v »parent-child« relaciji.

Relacijsko podatkovno bazo konstruiramo s pomočjo procesov normalizacije, ki odstranjujejo

redundanco in soodvisnost podatkov. Poznamo več stopenj normalizacije,[13] za relacijske podatkovne baze pa je priporočljivo, da so normalizirane vsaj do tretje normalne forme (3NF).

Slika 8 prikazuje normalizirano podatkovno shemo podatkovnega skladišča, izpeljano iz zvezdne sheme na sliki 7.



Slika 8: Normalizirano podatkovno skladišče

Prednosti normaliziranih podatkovnih skladišč so:

- v manjši velikosti v primerjavi z dimenzijskimi podatkovnimi skladišči,
- pri integriranju podatkov iz različnih izvornih sistemov nimamo redundantnih podatkov in posodabljanje podatkov poteka na enem mestu.

Za namene poslovnih analiz in poizvedb pa so dimenzijska podatkovna skladišča primernejša:

- ker so bolj enostavna (enonivojska navezava dimenzij na tabelo dejstev) – uporabniki hitreje razumejo podatkovni model,
- ker so poizvedbe hitrejše.

[13] Za več informacij o procesu normalizacije glej [20].

3.6 Zgodovina

Ena izmed bistvenih razlik med podatkovnim skladiščem in transakcijskimi sistemi je zmožnost podatkovnega skladišča, da hrani zgodovino.

Transakcijski sistemi zaradi potrebe po odzivnosti sistema navadno shranjujejo zgodovino prometnih podatkov nekaj let (od 3 do 5 let) in načeloma ne shranjujejo več verzij podatkov, ampak samo zadnjo verzijo. Kot verzije podatkov so tukaj mišljene različne napovedi (npr.: vremena, gibanja cen na borzi, gibanja porabe električne energije ...), ki jih izdelujemo vsak dan za izbrano obdobje v prihodnosti in so vsak dan različne. Transakcijski sistemi preteklih napovedi ne shranjuje, ampak jih posodabljajo z zadnjo (najbolj svežo) različico. Podatki, ki so starejši od pet let, se navadno shranijo v varnostne kopije oziroma v arhiv, ki je ločen od aktivnega transakcijskega sistema.

Transakcijski sistemi načeloma tudi ne shranjujejo vse zgodovine sprememb, ki se zgodijo na matičnih podatkih. Če v takem transakcijskem sistemu partner spremeni naslov ali naziv, bo spremenjeni naslov veljal za vso obstoječo zgodovino.

Pri podatkovnih skladiščih teh omejitev nimamo. Podatkovna skladišča lahko hranijo vso zgodovino prometa, vse verzije podatkov, vso zgodovino sprememb na matičnih podatkih s pomočjo »počasi spreminjajočih se dimenzij« (angl. *slowly changing dimension – SCD*), omogoča pa tudi t. i. posnetke (angl. *snapshots*) stanja transakcijskega sistema v določenem trenutku. Vse to omogoča izgradnjo kompleksnih modelov s pomočjo analitičnih orodij poslovne inteligence, pri katerih iščemo trende v podatkih, pregledamo evolucijo gibanja cen na evropskih borzah in izvajamo podatkovno rudarjenje.

Omenili smo t. i. »počasi spreminjajoče se dimenzije« (SCD), ki nam pomagajo obravnavati spremembe na matičnih podatkih. Gre za posebno vrsto dimenzij, katerih vrednosti atributov ostanejo večino časa enake, ko pa pride do spremembe, imamo na voljo več strategij obravnavanja spremembe. Kimball je bil prvi, ki je podal načine obravnavanja sprememb v dimenzijskih tabelah, kot jih poznamo danes [3]:

- **SCD Type 1**: pri tem tipu dimenzije nova vrednost atributa prepiše staro vrednost. Tip 1 v bistvu sploh ne shranjuje prejšnjih vrednosti atributa.
- **SCD Type 2**: pri tem tipu dimenzije nova vrednost atributa povzroči dodajanje novega zapisa v dimenzijo. V novi zapis prepišemo vse vrednosti nespremenjenih atributov ter spremenjeno vrednost atributa. Dimenzija, pri kateri je implementiran SCD Type 2, mora imeti definiran atribut, ki nam v primeru spremembe pove, kateri zapis v dimenziji vsebuje historične vrednosti atributov in kateri zapis trenutno veljavne.

Tipična implementacija dimenzije SCD Type 2 predvideva tri dodatne atribute v dimenziji, in sicer z naslednjimi lastnostmi:

  - Atribut z binarno zalogo vrednosti, ki nam pove, ali je zapis v dimenziji historičen (npr.: vrednost 0) ali pa trenutno veljaven (npr.: vrednost 1). Trenutno veljaven je lahko samo en zapis.
  - Dva datumska atributa, ki nam hranita informacijo o tem, od kdaj do kdaj je bil neki historični zapis veljaven. Trenutno veljaven zapis ima definirano, od kdaj velja, nima pa definiranega konca veljavnosti.

- **SCD Type 3**: pri tem tipu dimenzije nova vrednost atributa povzroči dodajanje novega atributa v dimenzijo. Novi atribut hrani prejšnjo vrednost atributa, medtem ko obstoječi atribut hrani najnovejšo vrednost.
- **SCD Type 4**: spremembe vrednosti atributa rešujemo z dodatno historično tabelo. Pri tem dimenzija shranjuje trenutno vrednost atributa, medtem ko dodatna historična tabela shranjuje vse spremembe vrednosti tega atributa.
- **SCD Type 6**: kombinira pristope tipov 1, 2 in 3 (1 + 2 + 3 = 6). Kimbal je ta tip imenoval »Unpredictable Changes with Single Version Overlay«[14] dimenzij.

3.7 Paketno osveževanje

Podatkovna skladišča se osvežujejo paketno s pomočjo ETL-sistema. Razlogov za paketno osveževanje je več:

- **Zagotavljanje konsistentnosti podatkov**. Podatkovno skladišče integrira podatke iz več izvornih sistemov, ki se neprestano spreminjajo. Če v več sistemih hranimo podatke (npr.: o količini prodane elektrike na trgu) in dovolimo, da se podatkovno skladišče osvežuje v realnem času, je izvajanje analiz oteženo, saj se številke neprestano spreminjajo.
- **Performančni vpliv na transakcijske sisteme**. Če bi imeli osveževanje podatkovnega skladišča v realnem času, bi lahko z neprestanimi zahtevami ETL-sistema po prenosu podatkov iz izvornih sistemov povzročili nižje performance transakcijskega sistema, saj bi le-ta moral, poleg »običajnega« dela, servisirati še zahteve podatkovnega skladišča.
- **Performančni vpliv na druga orodja podatkovnega skladišča**. Če podatkovno skladišče ponuja tudi večdimenzijske podatkovne baze tipa MOLAP,[15] ima neprestano osveževanje podatkov velik učinek na odzivnost in hitrost njenih poizvedb.

[14] Več o tem tipu dimenzij je napisano v [3].

[15] Več o vrstah OLAP-kock v poglavju 3.8.2.

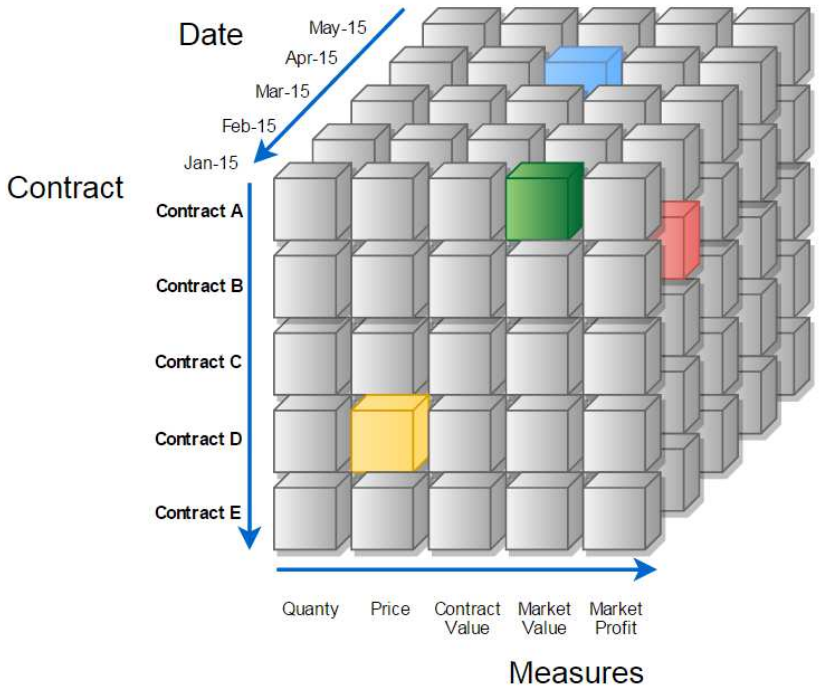
3.8 Orodja podatkovnega skladišča in OLAP-baze

Škoda je imeti veliko količino urejenih, konsistentnih, preverjenih, zgodovinskih podatkov v podatkovnem skladišču, če nimamo orodij, ki te podatke izkoristijo. Za dostop do podatkov v podatkovnem skladišču se uporabljajo različna orodja, kot so preglednice, vrtilne tabele (angl. *pivot tables*), orodja, specializirana za poročanje (SQL Server Reporting Services, Cognos, Crystal Reports, Tableau, Excel ...), orodja za analitiko (Pyramid Analytics, ProClarity, QlikView, Tableau ...) in orodja za podatkovno rudarjenje (IBM SPSS Modeler, Microsoft Analysis Services, SAS Enterprise Miner ...).

Nekatera izmed teh orodij uporabljajo poizvedbe SQL za dostop do podatkov, shranjenih v relacijski podatkovni bazi, nekatera pa delujejo na večdimenzijskih podatkovnih bazah, znanih tudi kot OLAP-kocke (angl. ***O**nL**ine** **A**nalytical **P**rocessing cubes*).

OLAP-kocka je posebna vrsta podatkovne baze, pri katerih so podatki shranjeni v celice, pozicijo vsake od njih pa definirajo spremenljivke – dimenzije. Vsaka celica predstavlja neki dogodek, vrednosti dimenzij pa povedo, kdaj in kje se je ta dogodek zgodil [17].

Slika 9 prikazuje primer tridimenzionalne OLAP-baze, z merami na osi X, dimenzijo *Contract* na osi Y in dimenzijo *Date*, ki nam v tem primeru predstavlja obdobje dobave električne energije na osi Z.



Slika 9: OLAP-baza

Z barvami označene posamezne celice pomenijo naslednje:

- Zelena: Tržna vrednost pogodbe A za obdobje dobave električne energije v januarju 2015.
- Rumena: Povprečna cena pogodbe D za obdobje dobave električne energije v januarju 2015.
- Modra: Vrednost pogodbe A za obdobje dobave električne energije v aprilu 2015.
- Rdeča: Dobiček pogodbe B za obdobje dobave električne energije v februarju 2015.

3.8.1 Razlike med OLAP-bazami in relacijskimi podatkovnimi bazami

Pri OLAP-bazah v nasprotju z relacijskimi podatkovnimi bazami poznamo:

- **Hierarhije** (angl. *Hierarchy*): Hierarhija je množica medsebojno odvisnih (*parent-child*) atributov v več nivojih, pri katerih atribut na najvišjem nivoju agregira vrednosti vseh podrejenih atributov. Najboljši primer hierarhije znotraj dimenzije je naravna hierarhija, sestavljena iz atributov leto-četrtletje-mesec-dan znotraj datumske dimenzije.
- **Agregacije** (angl. *Aggregations*): Poleg shranjevanja vrednosti posameznih celic v tabeli dejstev OLAP-baze gradijo tudi agregirne vrednosti iz teh celic, glede na atribute spremljevalnih dimenzij. Na voljo imamo več tipov agregacije (vsota, povprečje, utežno povprečje, delež ...), po katerih podatke agregiramo. Agregacija tipa vsota prek datumske dimenzije npr. sešteva celice po hierarhiji dan, mesec, četrtletje in leto.

Prednost OLAP-baz v primerjavi z relacijskimi podatkovnimi bazami je ravno v agregacijah in dodatnih izračunih, ki v kombinaciji z osnovnimi podatki v celicah omogočajo hitre odgovore na kompleksna vprašanja oziroma poizvedbe. V splošnem velja, da v primeru kompleksnih poizvedb OLAP-baza potrebuje samo majhen delež časa za odgovor v primerjavi z ekvivalentno poizvedbo na relacijski podatkovni bazi.

3.8.2 Vrste OLAP-baz

V načinu grajenja agregacij in hranjenju podatkov poznamo več tipov OLAP-baz:

- **MOLAP** (angl. ***M**ultidimensional **O**n**L**ine **A**nalysis **P**rocessing*): MOLAP je najpogostejši tip OLAP-baz. Pri tem tipu se vse agregacije predhodno izračunajo in se skupaj s podatki shranijo v posebej optimizirane večdimenzijske tabele, imenovane kocke. Kocke vsebujejo vse možne kombinacije pogledov/agregatov za neki podatek glede na število dimenzij in atributov, zato so posledično poizvedbe hitre. Slabost

MOLAP-baz je v njihovem procesiranju – grajenju agregatov, ki je lahko ob veliki količini podatkov dolgotrajno. Slabost MOLAP-baz je tudi v prikazovanju velikega števila tekstovnih opisov (t. i. tabelaričnega izpisa podatkov).

- **ROLAP** (angl. ***R**elational **O**n**L**ine **A**nalysis **P**rocessing*): Gre za alternativo MOLAP-u, pri katerem so podatki shranjeni samo v relacijski podatkovni bazi. Podatki in agregacije niso predhodno izračunani, ampak se izračunavajo »on-the-fly« med izvajanjem poizvedbe. Prednosti ROLAP-a v primerjavi z MOLAP-om so v odsotnosti procesiranja, boljši skalabilnosti pri obravnavanju velike količine podatkov in v dejstvu, da so podatki shranjeni v relacijski podatkovni bazi, ter s tem uporaba orodij in sistemov za upravljanje podatkovnih baz. Slabost ROLAP-a pa so počasnejše poizvedbe v primerjavi z MOLAP-om.
- **HOLAP** (angl. ***H**ybrid **O**n**L**ine **A**nalysis **P**rocessing*): Gre za »poroko« med MOLAP-a in ROLAP-om, pri kateri so podatki razdeljeni med multidimenzijsko in relacijsko podatkovno bazo. S tem pristopom želimo kombinirati hitrost poizvedb MOLAP-a in skalabilnost ROLAP-a. HOLAP-baze tako uporabljajo predhodno izračunane agregacije in podatke v večdimenzijskih tabelah do nekega nivoja ločljivosti (granulacije), pri višji ločljivosti (bolj podrobni) pa so podatki shranjeni v relacijski podatkovni bazi.

3.8.3 In-Memory OLAP

V prejšnjem poglavju smo omenili, da je hitrost izvajanja poizvedb nad neko množico podatkov ena izmed glavnih zahtev uporabnikov. Pri tem sta ključna dejavnika hitrost in velikost pomnilnika računalniškega sistema. Najbolj idealno bi torej bilo, če bi imeli na voljo hiter pomnilnik v (skoraj) neomejeni velikosti.

Toda v zgodovini, stroškovno gledano, sta si bili ti dve zahtevi v medsebojnem nasprotju. Hiter in velik pomnilnik je bil drag. Zato se je v računalniških sistemih uveljavila večnivojska pomnilniška hierarhija, ki jo sestavljajo majhni hitri predpomnilniki v hitrost blizu centralno procesne enote (predpomnilnik L1, L2), nekaj večji počasnejši glavni spomin in največji, vendar najpočasnejši pomožni pomnilnik (diski SSD, trdi diski, magnetni trakovi, diskovna polja ...).

Pri tradicionalnih OLAP-bazah (MOLAP, ROLAP, HOLAP) so podatki shranjeni v obliki večdimenzijskih ali relacijskih tabel na trdem disku. Čeprav današnji sistemi za upravljanje podatkov poskušajo minimizirati dostopni in bralni čas, potreben za pridobivanje podatkov z diska, je izvajanje kompleksnih poizvedb nad veliko količino podatkov časovno potratna operacija.

V zadnjih letih pa je prišlo do nekaj sprememb, ki so omogočile skoraj idealen scenarij – hiter in skoraj neomejen pomnilnik:

- prehod na 64-bitno arhitekturo, ki je razširil naslovni prostor delovnega spomina s 4 GB na več kot 100 GB,
- cenejša in zmogljivejša strojna oprema (večjedrni procesorji, hitrejši, večji ter močno cenejši pomnilniški moduli, paralelno procesiranje ...),
- pojav stolpično usmerjenih podatkovnih baz[16] (angl. *column-oriented databases*), boljših algoritmov za stiskanje podatkov ter novih mehanizmov za upravljanje agregiranih podatkov.

V »In-Memory« OLAP-baze so torej podatki shranjeni v stolpično usmerjeni OLAP-bazi v delovnem pomnilniku.

Osveževanje podatkov v »In-Memory« OLAP-bazi sicer poteka na podoben način kot pri MOLAP-u (branje podatkov iz relacijske podatkovne baze in shranjevanje le-teh v optimizirano podatkovno shemo v glavnem pomnilniki), vendar gre pri tem le za branje podatkov z diska, medtem ko procesiranje MOLAP-baze zahteva branje (iz relacijske podatkovne baze) z diska in zapisovanje na disk (v podatkovno bazo MOLAP).

Glavna slabost »In-Memory« OLAP-baz je omejitev velikosti baze na razpoložljivo velikost glavnega pomnilnika, druge omejitve pa so trenutno bolj posledica »mladosti« tehnologije kot pa tehnoloških omejitev.

[16] Microsoft je z »In-Memory« analitiko začel z vtičnikom PowerPivot v Excelu 2010. Pozneje so jo pod imenom Tabular vključili v Analysis Services SQL Serverja 2012. Oracle je lastno verzijo »In-Memory« analitike vpeljal v Oracle Database 12c v letu 2014.

4 Podatkovno skladišče v Skupini GEN-I

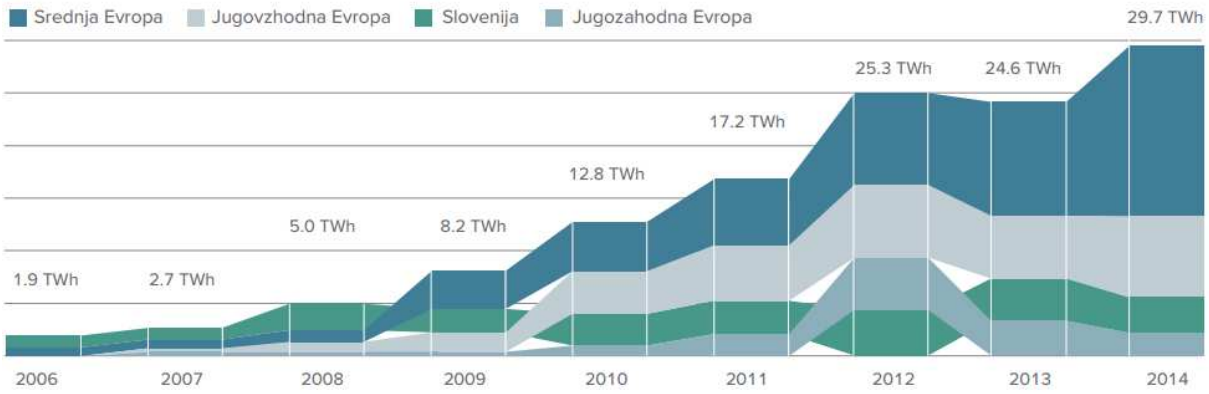
4.1 Stanje pred uvedbo podatkovnega skladišča v Skupini GEN-I

Začetki skupine GEN-I segajo v leto 2006, ko je skupina Gen Energija skupaj s podjetjem Istrabenz-Gorenje ustanovila hčerinsko družbo GEN-I, d. o. o., z namenom prodaje električne energije poslovnim odjemalcem na takrat že liberaliziranem trgu električne energije za poslovne odjemalce v Sloveniji in trgovanjem z električno energijo na trgih EU in JV Evrope. Sedež novoustanovljene družbe so postavili v Krško, lokacijo organizacijske enote za prodajo (*Prodaja*) končnim odjemalcem v Novo Gorico, medtem ko so lokacijo organizacijske enote za trgovanje (*Trgovanje*) z elektriko postavili v Ljubljano.

Na začetku je bila družba GEN-I, d. o. o. na Prodaji prisotna predvsem na slovenskem trgu ter v manjšem delu na italijanskem in avstrijskem trgu prodaje električne energije končnim poslovnim odjemalcem. Na Trgovanju je družba pokrivala predvsem nemški organizirani trg in v manjši meri sosednje trge (Avstrija, Italija, Madžarska). Zaradi različnih tržnih značilnosti na trgu na drobno in trgu na debelo ni bilo mogoče vzpostaviti enotnega informacijskega sistema za podporo Prodaji in Trgovanju, zato so se znotraj Prodaje in Trgovanja vzpostavile specializirane aplikacije (npr.: Prodaja končnim odjemalcem v tujini, Prodaja poslovnim odjemalcem v Sloveniji, Aplikacija za trgovanje ...).

Poleg teh specializiranih aplikacij so začele vznikati, tako na Prodaji kot na Trgovanju, tudi vzporedne ad hoc podatkovne baze za poročanje, za shranjevanje cen trgov, za tečajnice in druge, ki niso bile ne standardizirane ne primerno načrtovane in so v nekaterih primerih celo podvajale kakšno področje (npr.: cene trgov). Težave teh ad hoc podatkovnih baz je bila kvaliteta podatkov, saj v veliko primerih te baze niso imele določenega skrbnika, ki bi skrbel za osveževanje in pravilnost podatkov. Zaradi tega je prihajalo do primerov, ko je bilo znotraj družbe istočasno več nasprotujočih si poročil za iste dogodke.

V naslednjih letih je družba z liberalizacijo električnih trgov v JV in srednji Evropi hitro širila svoje trgovanje na te trge. Obseg poslovanja je strmo narastel, z njim pa je narasla tudi kompleksnost obvladovanja podatkov tako na področju shranjevanja kot tudi na področju zagotavljanja pravilnosti, pravočasnosti ter zajemu teh podatkov.



Slika 10: Rast prodaje električne energije Skupine GEN-I skozi leta [4]

V tem času pa ni rasla samo kompleksnost obvladovanja podatkov, ampak tudi želje in poslovne zahteve uporabnikov, ki so zaradi vedno bolj zapletenih razmer na evropskih trgih električne energije zahtevali nove analize, poročila in orodja, katerih razvoj in vzdrževanje sta zahtevala vedno več časa. Kmalu je prišlo do točke, ko je bilo z obstoječimi sistemi težko izpolnjevati te zahteve, oziroma nekaterih sploh ni bilo mogoče izpolniti.

Skupina GEN-I se je zgoraj opisanih razmer dobro zavedala in jih je želela z izgradnjo podatkovnega skladišča razrešiti. S podatkovnim skladiščem so želeli prečistiti in poenotiti podatke iz vseh ključnih virov in s področij, zagotoviti enoten vir za poslovno obveščanje, poročanje in analizo ter vzpostaviti enoten pregled nad Prodajo in Trgovanjem. Med dolgoročne cilje vzpostavitve podatkovnega skladišča je družba uvrstila uvedbo samopostrežne poslovne inteligence (angl. *Business Inteligence – BI*) in vzgojo naprednih uporabnikov-analitikov. Na ta način bi dosegla razbremenitev interne IT-službe, ki je velikokrat ozko grlo pri zagotavljanju podpore poslovnemu poročanju in analitiki.

4.2 O projektu

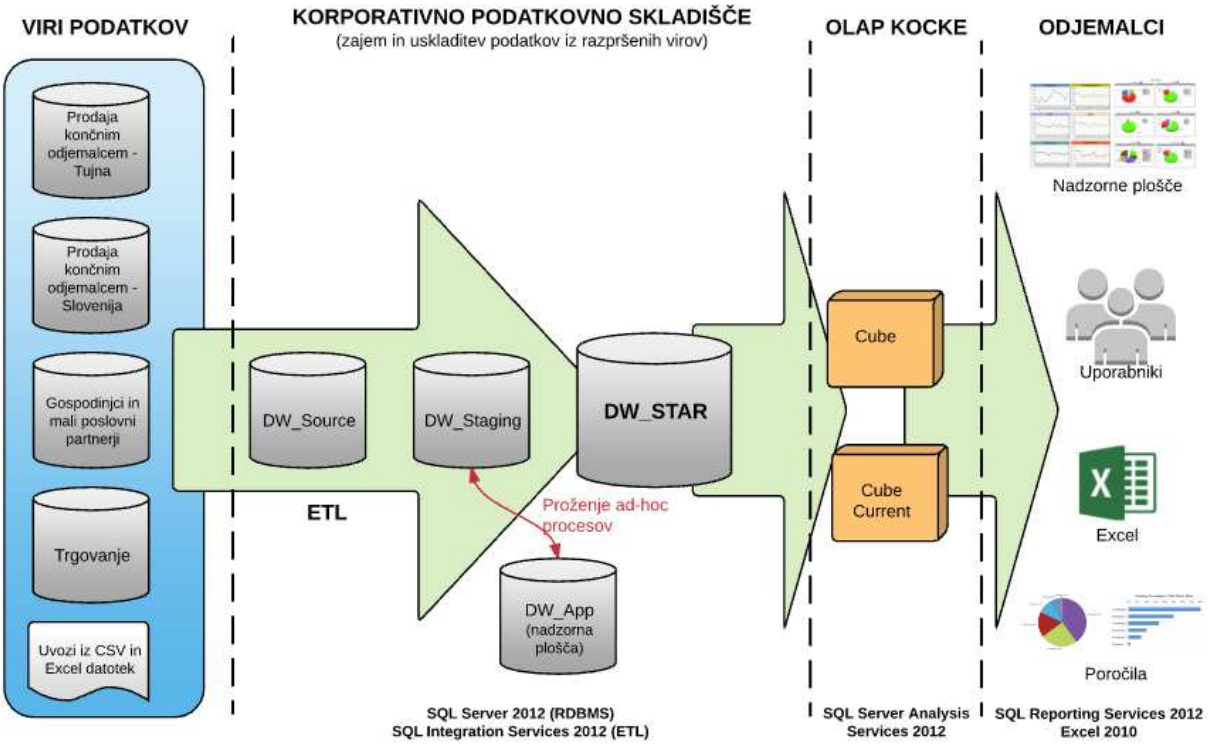
Začetki projekta segajo v leto 2010, ko smo v Skupini GEN-I začeli interno analizo o dostopnosti, kvaliteti in količini potrebnih podatkov za celovito poročanje. Analiza je pokazala, da je za zmožnost oblikovanja celovitega pogleda na poslovanje, kot tudi spremljanja in hitrega odzivanja na spremembe v poslovnem okolju priporočena izgradnja korporativnega podatkovnega skladišča. V tistem času v Skupini GEN-I ni bilo vzpostavljene ekipe, ki bi bila sposobna samostojno izpeljati tako zahteven projekt, zato se je Skupina odločila poiskati pomoč v obliki zunanjega svetovalca.

Oblikovala se je interna projektna skupina, katere član sem bil tudi sam, z nalogo poiskati primernega zunanjega svetovalca, ki bi poleg svetovanja opravil tudi prenos znanja ter dajal podporo pri izgradnji podatkovnega skladišča.

Izbor zunanjega izvajalca in ustrezne tehnologije je potekal skoraj celotno leto 2011 in na koncu smo se odločili za Microsoftovo BI platformo,[17] bazirano na Microsoftovem SQL Serverju 2012 – Enterprise Edition.

4.3 Arhitektura podatkovnega skladišča

Kot smo omenili v tretjem poglavju, so osnovni gradniki podatkovnega skladišča ETL-sistem, področje za preoblikovanje podatkov, področje za predstavitev podatkov in odjemalci. Podatkovno skladišče v Skupini GEN-I sledi tej arhitekturi z manjšimi prilagoditvami, kot je razvidno na sliki 11.



Slika 11: Arhitektura podatkovnega skladišča v Skupini GEN-I

Na področje za preoblikovanje podatkov spadata naslednji dve bazi:

[17] Več o komponentah Microsoftove BI platforme na http://www.microsoft.com/en-us/server-cloud/solutions/bi-analytics.aspx.

- **DW_Source**: v tej bazi se zbirajo podatki iz različnih virov. Črpanje podatkov iz virov mora imeti minimalen vpliv na izvorne sisteme, zato ETL-sistem črpa podatke v kar se da enaki obliki, kot so shranjeni v izvornih sistemih.
- **DW_Staging**: v tej bazi je shranjena vsa poslovna logika, na kakšen način moramo podatke transformirati. Baza vsebuje tudi pomožne tabele, če to zaradi optimizacije izvajanja zahteva proces transformacije podatkov.

Na področje za predstavitev podatkov spadata:

- **DW_Star**: kot že ime podatkovne baze sugerira, je v tej bazi shranjeno dimenzijsko podatkovno skladišče v zvezdni shemi.
- **OLAP-kocke**: znotraj SQL Server Analysis Services sta realizirani dve kocki tipa MOLAP. V poglavju 4.5 bomo bolj podrobno predstavili vsebino posamezne kocke.

Kot del podatkovnega skladišča se je vzpostavila tudi posebna baza DW_App, ki je namenjena podpornim in uporabniškim aplikacijam, s katerimi si poenostavljamo upravljanje podatkovnega skladišča.

4.4 Modeliranje podatkovnega skladišča

Modeliranje podatkovnega skladišča je potekalo po Kimballovem procesu modeliranja podatkovnega skladišča, ki je sestavljen iz štirih korakov [3]:

- Izbor poslovnega procesa, ki je osnova za model.
- Določanje podatkovne granulacije poslovnega procesa.
- Določanje dimenzij poslovnega procesa.
- Identificiranje mer poslovnega procesa.

4.4.1 Izbor poslovnega procesa

Kot osnovo/pomoč pri izboru poslovnega procesa nam je poslovodstvo Skupine določilo dve najpomembnejši poročili v skupini, Poročilo o stanju portfelja na Trgovanju in Poročilo o stanju portfelja na Prodaji.

S to omejitvijo smo začeli popisovati poslovne procese, udeležene pri obeh poročilih. Z izdelanim popisom poslovnih procesov smo nato določili ključne uporabnike obeh poročil in skupaj z njihovo pomočjo začeli zbirati poslovna vprašanja. Zelo hitro se je izkazalo, da je zaradi fizične ločenosti organizacijskih enot za Trgovanje in Prodajo prihajalo do različnih pojmovanj istih poslovnih dogodkov oziroma da so različni poslovni dogodki bili označeni z enakim pojmom. Naša prva je tako bila vzpostavitev enotne terminologije, da so se najprej

ključni uporabniki med seboj sploh razumeli in smo šele pozneje šli na njihove poslovne zahteve.

Kmalu smo sestavili seznam poslovnih zahtev, v katerih so uporabniki želeli:

- imeti podatke o gibanju cenovnih krivulj na različnih trgih električne energije v in zunaj Evrope,
- imeti podatke o vrednosti in dobičkonosnosti portfelja v evrski ali lokalni valuti,
- imeti vzpostavljen skupen pogled na pogodbe Trgovanja in Prodaje z namenom izdelave ocene finančne izpostavljenosti do izbranega partnerja,
- kategorizirati posamezne pogodbe po hčerinskih podjetjih,
- imeti vrednost pogodbe v lokalni in evrski valuti skupaj
- ter še mnogo drugih.

S seznama poslovnih vprašanj smo nato izluščili tri področja, ki so nam predstavljala tri tabele dejstev:

- *Pogodbe – FactContract,*
- *Cenovne krivulje – Fact PriceCurve,*
- *Tečajnice – FactExchangeRate.*

4.4.2 Določanje podatkovne granulacije

Pri določanju podatkovne granulacije poslovnega procesa smo upoštevali vodilo dobre prakse, po katerem je treba dimenzijsko podatkovno skladišče zgraditi okoli čim bolj atomarnih podatkov, ki so na voljo v izvornih sistemih za izbrani poslovni proces [3]. S tem vodilom smo bili usklajeni z zahtevami poslovnih uporabnikov, ki so si tudi želeli imeti podatke na najnižjem možnem nivoju.

Kot smo navedli v poglavju 2.42.3, je obračunski interval električne energije in s tem pogodbe v izvornem sistemu enak eni (1) uri. Zaradi tega smo za tabeli dejstev *FactContract* in *FactPriceCurve* določili urno podatkovno granulacijo.

Po drugi strani so bili podatki o tečajnicah v izvornih sistemih na voljo na dnevnem nivoju, zato smo za *FactExchangeRate* določili dnevno podatkovno granulacijo.

4.4.3 Določanje dimenzij poslovnega procesa

Pri določanju dimenzij posamezne tabele dejstev smo si pomagali s t. i. bus matriko. Gre za orodje, pri katerem imamo v vrsticah naštete tabele dejstev, v stolpcih pa dimenzije, po katerih poizvedujemo/analiziramo/filtriramo podatke v tabelah dejstev. Slika 12 prikazuje osnovno bus matriko.

	DIMENZIJE									
TABELE DEJSTEV	Date	Time	Contract	PriceCurve	ExchangeRate	Currency	Subsidiary	Partner	Market	SnapshotDate
FactContract	x	x	x			x	x	x	x	x
FactPriceCurve	x	x		x		x			x	x
FactExchangeRate	x				x	x				x

Slika 12: Bus matrika

Kot je razvidno s slike 12, je tabela dejstev *FactExchangeRate* povezana z dimenzijo *Date*, *ExchangeRate*, *Currency* in *SnapshotDat*e, tabela dejstev *FactPriceCurve* z dimenzijo *Date*, *Time* in tako naprej.

Na sliki tudi opazimo, da so nekatere dimenzije (*Date*, *Time*, *Currency* in *SnapshotDate*) skupne vsem trem področjem – gre za t. i. skupne dimenzije (angl. *Conformed Dimensions*). Poznamo sicer več vrst skupnih dimenzij,[18] vendar bomo za svoje potrebe uporabljali najosnovnejšo vrsto, kjer so skupne dimenzije enake v različnih tabelah dejstev [3].

Pri tem je treba poudariti, da gre pri zgornji bus matriki za generaliziran pregled, saj nekatere dimenzije nastopajo v več različnih vlogah. Na primer dimenzija *Date* na področju *Pogodb* nastopa v vlogi obdobja dobave, datuma izdaje računa in datuma plačila računa, medtem ko dimenzija *Currency* nastopa v vlogi kot valuta sklenjenega posla, valuta evaluacije in valuta plačila računa.

Časovni dimenziji *Date* in *Time* bomo bolj podrobno opisali v poglavju 5, v nadaljevanju pa so podani opisi drugih dimenzij:

- **Contract**: Dimenzija hrani lastnosti pogodbe. Naštevanje vseh atributov dimenzije presega namen te naloge, zato bomo navedli samo nekaj najpomembnejših atributov:

[18] Več o različnih vrstah skupnih dimenzij v [3].

- *BuySell*: Atribut nam pove, ali gre za nakupno ali prodajno pogodbo. V odvisnosti od vrednosti tega atributa imamo pozitivno ali negativno količino električne energije v *FactContract*.
- *ContractType*: Atribut nam pove, kakšnega tipa je pogodba. Kot smo spoznali v poglavju 0, imamo lahko pogodbo s fizično dobavo, pogodbo z denarno poravnavo in pogodbe s kapacitetami.
- *ContractDate*: Atribut nam pove datum sklenitve posla. Čeprav gre za datumsko polje in bi lahko iz njega izpeljali datumsko dimenzijo, smo se odločili, da to ostane lastnost pogodbe. Razlog za to odločitev je bil predvsem v nezdružljivosti z drugimi področji (Cenovne krivulje in Tečajnice).
- *DeliveryPeriodFrom*: Začetek obdobja dobave električne energije.
- *DeliveryPeriodTo*: Konec obdobja dobave električne energije.
- *BusinessUnit*: Atribut nam pove, katera organizacijska enota je sklenila posel.
- *ContractTimeZone*: Kot že samo ime namiguje, nam atribut pove, v kateri časovni coni je bila pogodba sklenjena.
- *ContractSID*: Ključ dimenzije.

- **PriceCurve**: Dimenzija hrani lastnosti cenovne krivulje. Nekaj najpomembnejših atributov:
  - *PriceCurveShortName*: področje cenovnih krivulj se veliko uporablja v različnih grafih, zato ima vsaka cenovna krivulja kratko ime oziroma kratico.
  - *PriceCurveLongName*: dolgo ime cenovne krivulje je standardizirano. Generiranje imena je podvrženo pravilom poimenovanja z namenom, da uporabnik že iz imena krivulje natančno ve, za katero dobrino, za kateri tip pogodbe in za kateri trg (*Market*) velja cenovna krivulja.
  - *PriceCurveTimeZone*: Tako kot pri pogodbah imamo tudi pri cenovnih krivuljah atribut, ki nam pove časovno cono, za katero velja cenovna krivulja.
- **ExchangeRate**: Dimenzija hrani lastnosti tečajnic. Nekaj najpomembnejših atributov:
  - *ExchangeRateType*: Atribut nam pove tip tečajnice, in sicer ali gre za srednji, nakupni ali prodajni tečaj.
  - *CurrencyForCurrency*: atribut nam pove, kateri dve valuti nastopata v tečajnici.
  - *Bank*: Atribut nam pove, od katere banke je tečajnica.
- **Partner:** Dimenzija hrani lastnosti partnerjev. Nekaj najpomembnejših atributov:
  - *PartnerName*: uradni naziv partnerja.
  - *VAT Number*: davčna številka partnerja. Atribut je pomemben pri procesu identifikacije podvojenih partnerjev in zagotavljanju enotne slike partnerja v podatkovnem skladišču.
  - *Country*: Država, v kateri ima partner sedež. Atribut je pomemben pri določanju, ali se DDV obračuna ali ne.

- **Market**: Dimenzija hrani lastnosti električnih trgov. Nekaj najpomembnejših atributov:
    - *DefaultMarketPrice*: vsak trg ima določeno privzeto cenovno krivuljo, po kateri se vrednotijo vse pogodbe, ki so sklenjene na tem trgu.
    - *MarketRegion*: atribut nam podaja območje veljavnosti trga. Načeloma velja, da ima ena država en trg električne energije, vendar pa to ni nujno res. V poglavju 2 smo spoznali, da je namen liberalizacije trgov z električno energijo tudi vzpostavitev enotnega trga znotraj EU. Najbližje temu cilju sta trga Avstrije in Nemčije, kjer je že danes cena električne energije enaka. V tem primeru nam regija trga zajema Nemčijo in Avstrijo.
- **Subsidiary**: Kot smo zapisali v poglavju 4.1, Skupina GEN-I nastopa na skoraj vseh trgih električne energije v Evropi. Na nekaterih trgih električne energije v EU lahko Skupina nastopa z matično slovensko družbo, medtem ko mora na drugih trgih nastopati z lokalno hčerinsko družbo. Zaradi tega mora imeti vsaka pogodba, ki jo sklene Skupina, označbo, katera hčerinska ali matična družba je sklenila ta posel. Dimenzija Subsidiary tako hrani lastnosti matične in hčerinskih družb. Nekaj najpomembnejših atributov dimenzije:
    - *SubsidiaryShortName*: atribut nam podaja uradni kratki naziv matičnega/hčerinskega podjetja.
    - *DefaultInCountry*: v neki državi je lahko možnih več hčerinskih podjetij Skupine GEN-I in ta atribut nam posreduje informacijo, katero izmed teh hčerinskih podjetij je privzeto za izbrano državo.
- **Currency**: Dimenzija hrani lastnosti valute. Kot vemo iz poglavja 5.1, je Skupina prisotna tudi na trgih, kjer uradna valuta ni EUR, ampak lokalna valuta. Tako mora imeti vsaka pogodba navedeno, v kateri valuti je bila pogodba sklenjena in v kateri valuti mora biti pogodba plačana. Nekaj najpomembnejših atributov dimenzije:
    - *CurrencyCode*: tričrkovna ISO standardna oznaka valute.
    - *CurrencyName*: angleški naziv valute.
    - *CurrencyNumber*: številčna ISO standardna oznaka valute.
    - *InUse*: atribut nam pove, ali se valuta uporablja v podatkovnem skladišču ali ne.

4.4.4 Identificiranje mer poslovnega procesa

Pri identifikaciji mer za tabele dejstev *FactContract*, *FactPriceCurve* in *FactExchangeRate* smo se znašli pred naslednjimi izzivi:

- Kako poenotiti in konsolidirati različna poimenovanja mer v obstoječih poročilih?
- Kako zagotoviti konsistentno in standardizirano poimenovanje mer, iz katerih bo uporabnik že iz imena sklepal, za kakšno vrsto mere gre?
- Kako zagotoviti edinstveno ime mere?[19]

Za razrešitev teh težav smo vzpostavili interni pravilnik poimenovanja mer, v katerem so se podrobno navedla pravila za generiranje imen.

Pri identifikaciji mer izbranega poslovnega procesa je prav tako dobro vedeti, ali so posamezne mere aditivne ali ne. Aditivne mere namreč lahko seštevamo po različnih atributih posameznih dimenzij in je dobljena vsota pravilen rezultat; če bi na podoben način seštevali neaditivne mere, bi dobili nesmiselni rezultat.

S pomočjo internega pravilnika poimenovanja mer smo pripravili seznam mer, razdeljenih po tabelah dejstev z naslednjimi atributi:

- Področje (tabela dejstev).
- Naziv mere v relacijski podatkovni bazi.
- Formula izračuna mere v relacijski podatkovni bazi.
- Indikator na relacijski podatkovni bazi, ki pove, ali gre za osnovno mero (O), izračunano mero iz osnovnih (I), ali pa pomožno mero, ki uporabniku ni dostopna in je potrebna za izračun neke druge mere (V).
- Naziv mere v OLAP-kocki.
- Formula izračuna mere v podatkovni bazi OLAP.
- Indikator na podatkovni bazi OLAP, ki pove, ali gre za osnovno mero ali izračunano mero iz osnovnih mer.

V nadaljevanju so podane tabele dejstev z nekaterimi merami glede na zgoraj navedene atribute.

4.4.4.1 Pogodbe (FactContract)

V tabeli dejstev *FactContract* so shranjene vse pogodbe, ki se sklenejo pri trgovanju in prodaji z električno energijo na trgih električne energije.

[19] Zahteva po edinstvenosti imen mer je posledica tehnološke omejitve Analysis Services 2012, kjer se zahteva edinstvenosti imen mer znotraj posamezne kocke, sicer kocke ni mogoče postaviti in tudi ne procesirati.

Tabela 1: FactContract

Naziv v SQL	Indikator	Formula	Naziv v OLAP	Indikator	Formula	Opomba
QtyMWh	O	/	Qty MWh	O	/	Količina električne energije v MWh
ContractPrice	O	/	WAvg Contract Price	I	Contract Value / Qty MWh	Utežena povprečna cena pogodbe v MWh
MarketPrice	O	/	WAvg Market Price	I	Market Value / Qty MWh	Utežena povprečna cena na trgu v MWh
ContractValue	I	QtyMWh * Price	Contract Value	O	/	Vrednost pogodbe
MarketValue	I	QtyMWh * MarketPrice	Market Value	O	/	Vrednost pogodbe na trgu
MarketProfit	/	/	Market Profit	I	Market Value - Contract Value	Dobiček/izguba pogodbe

Kot lahko razberemo iz Tabele 1, je mera MarketPrice v *FactContract* v relacijski podatkovni bazi osnovna mera, v OLAP-modelu pa se izračuna. Vzrok za tak izračun mere je v neaditivnosti mere in v tem, da gre pri tem še za utežno povprečje. Formula nam v tej obliki zagotavlja pravilen izračun tudi v primeru različnih nivojev časovne ali pogodbene hierarhije (npr.: smo na mesečnem/četrtletnem/letnem nivoju), oziroma kot je navedel Kimball:

»Pri neaditivnih merah je treba v tabeli dejstev shraniti števec in imenovalec ter nato neposredno v uporabniškem orodju za dostop do podatkov izračunavati mero kot utežno povprečje vsot, in ne kot vsoto utežnih povprečij. [3]«

Prav tako vidimo, da mera MarketProfit obstaja samo v OLAP-modelu, in ne v relacijski podatkovni bazi. Kimball sicer priporoča, da se takšni preprosti izračuni izvedejo že na relacijskem nivoju [3], vendar smo se odločili drugače zaradi naslednjih razlogov:

- Velike količine podatkov.[20]
- Zaradi prvega razloga smo želeli imeti na relacijskem modelu samo osnovne mere, ki bodo osnova vsem drugim meram. Vse ostale mere se realizirajo na OLAP-nivoju.
- Konsistentnost, kje so definicije mer. Uporabnik »ve«, da so v relacijski podatkovni bazi na voljo samo osnovne in preprosto izračunljive mere, vse druge (specializirane) mere pa so na voljo v OLAP-modelu.
- Morebitno kasnejše dodajanje nove mere, izpeljane iz osnovnih mer, bi v primeru vodenja teh mer v relacijski bazi povzročilo časovno potratno reindeksacijo (DROP in RECREATE INDEX) največje tabele v podatkovnem skladišču.[21]

4.4.4.2 Cenovne krivulje (FactPriceCurve)

V tabeli dejstev *FactPriceCurve* shranjujemo cene električne energije na trgih električne energije. Poleg uradnih cen so v tej tabeli shranjene tudi interne cene, izračunane po določenih formulah iz uradnih cen, ter cenovne razlike (angl. *spread*) med cenami trgov. Cene shranjujemo v originalni valuti, kot so na voljo na posameznem trgu in preračunane v valuto EUR.

Tabela 2: FactPriceCurve

Naziv v SQL	Indikator	Formula	Naziv v OLAP	Indikator	Formula	Opomba
Price	**O**	**/**	**Avg Price**	**O**	**/**	**Gre za ceno v originalni valuti**
Price_EUR	**I**	**Price * ExchangeRate**	**Avg Price EUR**	**O**	**/**	**Gre za ceno v valuti EUR**

4.4.4.3 Tečajnice ExchangeRate

V tabeli dejstev *FactExchangeRate* shranjujemo tečajnice.

Tabela 3: FactExchangeRate

Naziv v SQL	Indikator	Formula	Naziv v OLAP	Indikator	Formula	Opomba
ExchangeRate	**O**	**/**	**Avg Exchange Rate**	**O**	**/**	

[20] Več o količini podatkov v poglavju 6.

[21] Več o indeksaciji v poglavju 6.3.1.

4.4.4.4 Zakaj samostojni področji Cenovne krivulje in Tečajnice

Kot je razvidno iz Tabel 2 in 3, sta tabeli dejstev *FactPriceCurve* in *FactExchangeRate* dokaj preprosti, saj skupaj vsebujeta samo tri mere. Načeloma bi ju lahko zato vključili v tabelo dejstev *FactContract*, vendar smo se odločili za samostojni področji, ker:

- Mere področja Cenovnih krivulj in Tečajnic uporabljajo še kje drugje, in ne samo na področju *Pogodb*.
- Politika shranjevanja posnetkov stanja se za ti dve področji razlikuje od politike shranjevanja posnetkov stanja za *Pogodbe*.[22]

4.4.5 Generiranje dimenzij

Dimenzije podatkovnega skladišča lahko, glede na način njihovega osveževanja, razdelimo na tri skupine:

- **Statične**: Gre za dimenzije, ki se ne spreminjajo v življenjski dobi podatkovnega skladišča in se ne posodabljajo med rednim osveževanjem podatkovnega skladišča. Zapise v teh dimenzijah lahko generiramo avtomatsko z ustreznim algoritmom. V našem primeru sta to dimenziji *Time* in *Date*.
- **Skoraj statične**: Gre za dimenzije, ki se skoraj ne spreminjajo v življenjski dobi podatkovnega skladišča in se ne posodabljajo med rednim osveževanjem podatkovnega skladišča. Začetno stanje dimenzij moramo napolniti ročno (z ročnim vnosom v tabelo oziroma z vnaprej pripravljeno skripto), prav tako je ročno nadaljnje vzdrževanje dimenzij.[23] V našem primeru gre za dimenzije *Market, Subsidiary* in *Currency*.
- **Dinamične**: Gre za dimenzije, ki se posodabljajo med rednim osveževanjem podatkovnega skladišča s pomočjo ETL-procesa. V našem primeru gre za dimenzije *Contract*, *PriceCurve*, *ExchangeRate*, *Partner* in *SnapshotDate*.

4.4.6 Obravnava NULL vrednosti dimenzijskega ključa v tabeli dejstev

V mnogo tehničnih disciplinah obstajajo t. i. modra pravila in rdeča pravila. Modra pravila so splošna priporočila in smernice, ki veljajo v veliki večini primerov, vendar pa jih lahko

[22] Več o politiki shranjevanja v poglavju 7.

[23] Načeloma lahko problem vzdrževanja nekaterih kategorij (npr.: matični podatki partnerjev, držav, valut, geografskih podatkov ...) teh dimenzij rešujemo s pomočjo sistema MDM, ki pridobiva podatke iz javno dostopnih bazah. Kljub temu moramo še vedno ročno vzdrževati podatke, ki so pomembni poslovnim uporabnikom, v javno dostopnih bazah pa ti podatki niso na voljo.

prilagajamo oziroma »kršimo«, če situacija to od nas zahteva. Rdeča pa so tista pravila, ki jih ne smemo nikoli kršiti.

Pri podatkovnih skladiščih je eno izmed takih rdečih pravil (ne)shranjevanje NULL vrednosti dimenzijskega ključa v tabeli dejstev. Nikoli in nikdar se nam ne sme primeriti situacija, pri kateri imamo v tabeli dejstev vrednost dimenzijskega ključa NULL.

Poglejmo si scenarij, v katerem imamo tabelo dejstev *FactContract*, kjer hranimo podatke o količini električne energije po dimenzijah *Partner*, *Currency* in izpeljano datumsko dimenzijo *DueDate* (dimenzija nam daje informacijo o roku plačila računa). V veliki večini primerov, recimo v 98 %, imamo definiran rok plačila, v 2 % pa roka plačila nimamo definiranega – predpostavimo, da interni posli znotraj Skupine GEN-I nimajo specificiranega roka plačila.

Pri transakcijskih sistemih, ki niso prilagojeni poročanju in analiziranju, shranjevanje NULL vrednosti za ključ dimenzije ni težava in je v nekaterih primerih celo ustrezna rešitev. Pri podatkovnih skladiščih pa je shranjevanje NULL vrednosti dimenzijskih ključev slaba ideja.

Predpostavimo, da imamo poslovnega uporabnika, ki si želi narediti analizo količine električne energije po plačilnih rokih. Obstoj NULL vrednosti tujega ključa za dimenzijo *DueDate* v tabeli dejstev *FactContract* bi poslovnemu uporabniku pošteno otežil analizo, saj bi moral v količine obstoječih plačilnih rokih vključiti še količino NULL plačilnega roka. V nekaterih orodjih (npr.: Excel), je namreč treba posebej vklopiti možnost prikazovanja vrstic z NULL vrednostmi, hkrati pa mora uporabnik sam poskrbeti, da upošteva samo relevantne podatke z NULL vrednostmi.

Prav tako je v primeru dovoljevanja NULL vrednosti tujega ključa v tabeli dejstev poslovni uporabnik postavljen pred situacijo, v kateri ne ve, ali je NULL pravilna vrednost ali pa gre pri tem mogoče za napako v ETL-procesu.

Splošna praksa za razreševanje tega problema je uvedba enega ali več posebnih zapisov v dimenzijo, na katere navežemo prometne podatke z NULL vrednostmi tujih ključev pri zapisovanju le-teh v tabelo dejstev.

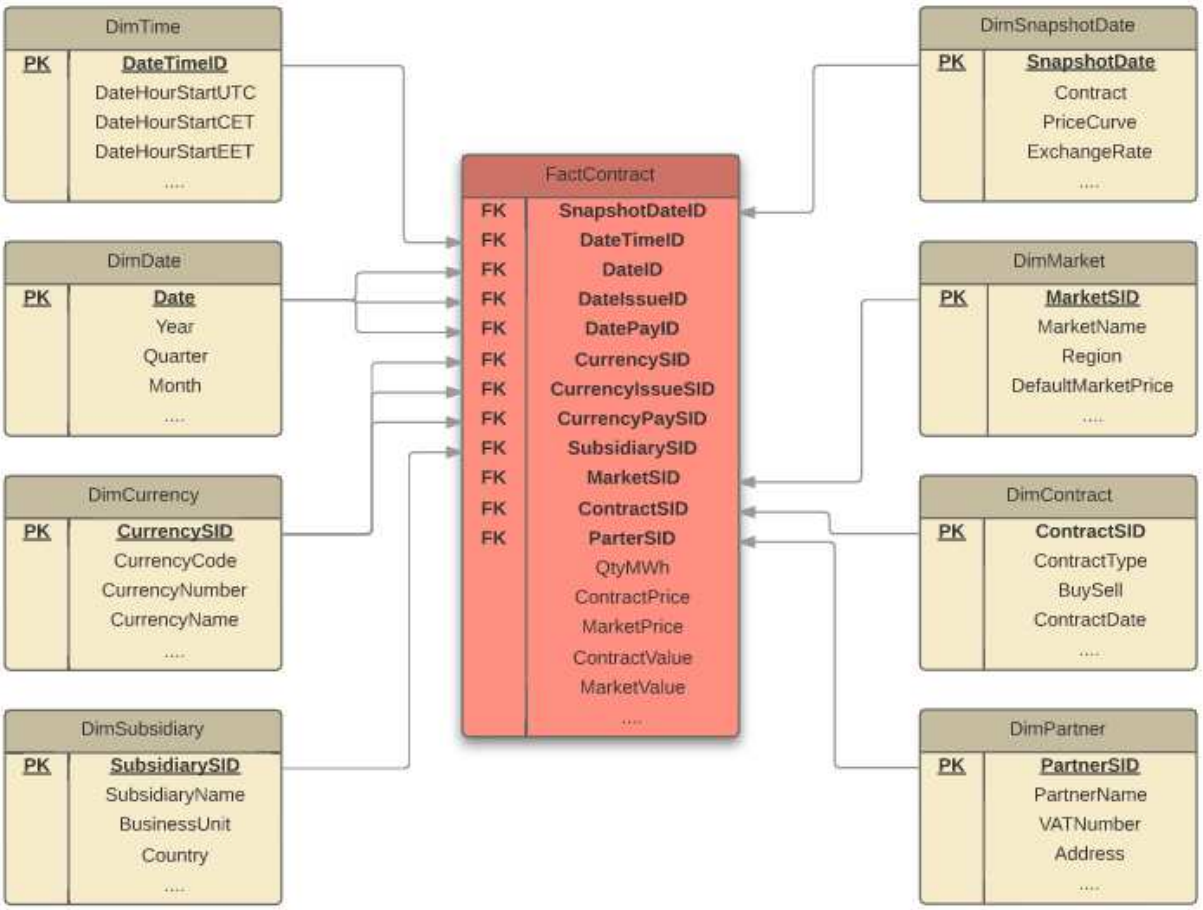
V našem primeru smo zato v vsako dimenzijo dodali dva posebna zapisa z lastnim nadomestnim ključem in vrednostjo:

- **Unknown**: Veljavno stanje prometnega podatka v izvornem sistemu.
- **Error during ETL process**: Pri osveževanju podatkovnega skladišča je prišlo pri tem prometnem podatku do napake.

4.4.7 Zvezdna shema

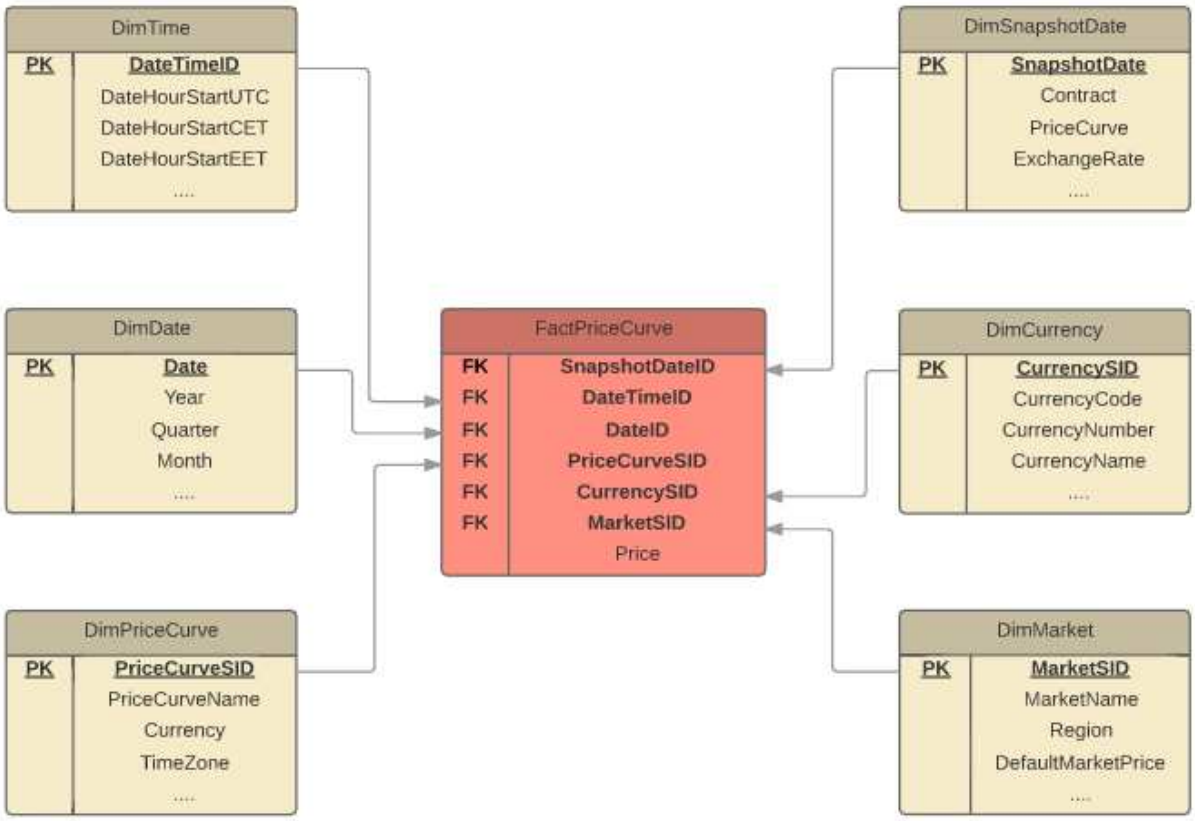
Za konec tega poglavja pa si oglejmo še končne zvezdne sheme vsakega od področij, glede na bus matriko v poglavju 0.

Področje Pogodbe



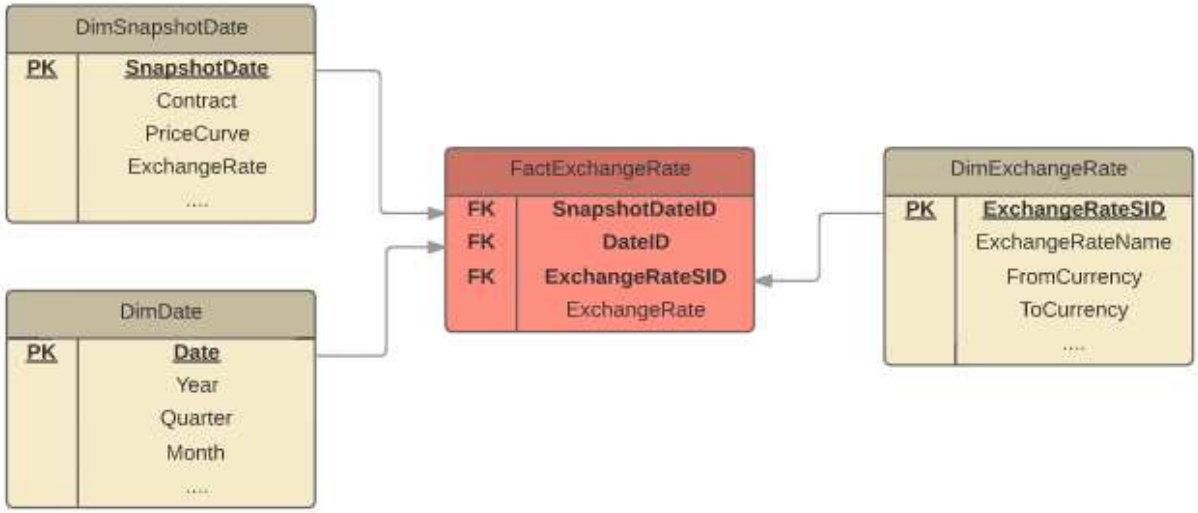
Slika 13: Zvezdna shema področja Pogodbe

Področje Cenovne krivulje



Slika 14: Zvezdna shema področja Cenovne krivulje

Področje Tečajnice



Slika 15: Zvezdna shema področja Tečajnice

4.5 Modeliranje sheme MOLAP-kocke

V poglavju 5.3 smo pri arhitekturi podatkovnega skladišča navedli, da imamo znotraj podatkovnega skladišča realizirani dve analitični podatkovni bazi MOLAP, Cube in Cube_Current. Gre za strukturno enaki analitični podatkovni bazi, ki se razlikujeta le v količini podatkov posameznega področja.[24] V nadaljevanju bom zato predstavil strukturo analitične podatkovne baze samo za Cube.

Vsaka analitična podatkovna baza v Analysis Services vsebuje naslednje analitične objekte:

- Povezave na podatkovne vire (angl. *Data Sources*) – vsebuje definicije, kateri so naši viri podatkov.
- Poglede na podatkovne vire (angl. *Data Source Views*) – vsebuje definicije, kako pridobivamo podatke iz podatkovnih virov.
- Kocke (angl. *Cubes*) – vsebujejo definicije kock, mere, kalkulacij, particij, agregacij, perspektiv …
- Dimenzije – vsebuje definicijo dimenzij.
- Strukture podatkovnega rudarjenja (angl. *Mining Structures*) – vsebuje definicije struktur podatkovnega rudarjenja.

V našem primeru smo definirali samo objekte kocke in dimenzije.
Pri tem smo se oprli na bus matriko iz poglavja 5.4.3, na zvezdne sheme posameznih področij iz poglavja 5.4.6. ter na identificirane mere iz poglavja 5.4.4.

Najprej smo v analitično podatkovni bazi definirali dimenzije:

- Date.
- Time.
- Contract.
- PriceCurve.
- ExchangeRate.
- Currency.
- Subsidiary.
- Partner.
- Market.
- SnapshotDate.

[24] Več v poglavju 8.

V naslednjem koraku smo v analitično podatkovno bazo dodali kocko Cube ter ji dodali naslednje skupine mer:

- Contract Electricity (navezava na tabelo dejstev *FactContract*).
- Price Curve (navezava na tabelo dejstev *FactPriceCurve*).
- Exchange Rate (navezava na tabelo dejstev *FactExchangeRate*).

V zadnjem koraku pa smo, glede na bus matriko, posameznim skupinam mer dodali skupne in izpeljane dimenzije. S tem smo v MOLAP-kocki efektivno združili na relacijski bazi ločene tabele dejstev in tako uporabniku prek skupnih dimenzij omogočili enostavne skupne analize teh področij.

5 Problematika obravnave časa

Obravnavanje časa je ena izmed najpomembnejših nalog pri izgradnji podatkovnega skladišča. Gre namreč za dimenzijo (ali več njih), ki je skoraj zagotovo vsebovana v vseh tabelah dejstev (oziroma v vseh področnih podatkovnih skladiščih) podatkovnega skladišča.

Pri obravnavi tega problema smo se srečevali z različnimi pogledi uporabnikov na isti poslovni dogodek, kot so različna časovna granulacija (mesečni pogled proti dnevnemu proti urnemu), upoštevanje časovnih con, neupoštevanje časovnih con, upoštevanje poslovnega dneva in drugih.

V nadaljevanju bomo tako predstavili nekatere vidike obravnavanja časa v podatkovnem skladišču, ki so doprinesli h grajenju časovne in datumske dimenzije.

5.1 Zapis poslovnega dogodka v podatkovnem skladišču

Kot smo spoznali v 3. poglavju, trgovanje na trgih električne energije poteka na urnem nivoju, kar pomeni, da lahko trgovec spreminja količino in ceno na pogodbi v vsakem od 24-urnih blokov. Poslovni dan električne energije torej lahko opišemo s časovnimi intervali, ki so prikazani v tabeli 4:

Tabela 4:

Urni blok	Urni blok – začetni interval	Urni blok – končni interval
00.00–01.00	0	1
01.00–02.00	1	2
02.00–03.00	2	3
....	...	...
23.00–00.00	23	24

Kot lahko razberemo iz tabele, lahko posamezni urni blok označimo z začetnim ali pa končnim intervalom.

Ta dvojnost zapisa posameznega urnega bloka se je kazala tudi v izvornih aplikacijah podatkovnega skladišča, saj so aplikacije za prodajo električne energije zapisovale prometne podatke na začetek urnega bloka, medtem ko je aplikacija za trgovanje zapisovala prometne podatke na konec urnega bloka.

Da bi bila zadeva še bolj zapletena, so bili prometni podatki v prodajnih aplikacijah zapisani v času UTC[25], medtem ko so bili v trgovalni aplikaciji zapisani v nekem nestandardiziranem internem času, imenovanem MET (Middle Europe Time), za katerega se je pozneje izkazalo, da gre v bistvu za čas UTC + 1 ura.

Pri zapisovanju prometnih podatkov v podatkovno skladišče je sicer že prej obstajal konsenz, da bodo le-ti v UTC-ju, vprašanje je ostalo samo, ali se bodo zapisovali na začetek ali na konec urnega bloka. Zaradi večje količine podatkov v prodajnih aplikacijah in posledično manjšega števila časovnih pretvorb med prenosom podatkov iz izvornih sistemov v podatkovno skladišče, smo se nazadnje odločili za zapis prometnih podatkov na začetek urnega bloka.

Pred seboj smo imeli še eno odločitev, in sicer ali naj se uvede poseben nadomestni primarni ključ časovne dimenzije celoštevilčnega podatkovnega tipa (angl. *Integer*) ali pa ga ohrani kot datumski podatkovni tip. Kimball sicer priporoča uvedbo posebnega primarnega ključa celoštevilčnega tipa [3], ker naj bi se na ta način prihranilo nekaj diskovnega prostora v primerjavi z datumskim podatkovnim tipom.[26] Prav tako naj bi bile poizvedbe z uporabo numeričnega primarnega ključa hitrejše v primerjavi z datumskim podatkovnim tipom, vendar tega nekateri poskusi niso pokazali [19].

Nazadnje smo se odločili za primarni ključ podatkovnega tipa *SmallDateTime*, ki je enake velikosti kot podatkovni tip *Integer* (4 bajte). Zaloga vrednosti podatkovnega tipa je definirana od 1. 1. 1900 do 6. 6. 2079, kar zadostuje potrebam podatkovnega skladišča.

Začetna polja dimenzije Time so prikazana v Tabeli 5:

Tabela 5

Naziv polja	Podatkovni tip v relacijski bazi	Opomba
DateTimeID	SmallDateTime	Ključ dimenzije, ki je enak polju DateHourStartUTC – zapis urnega bloka na začetek intervala
DateHourStartUTC	SmallDateTime	Odločili smo se za podvojitev polja zaradi »enakopravnosti« z ostalimi časovnimi conami
DateHourEndUTC	SmallDateTime	DateHourEndUTC – zapis urnega bloka na konec intervala

[25] Universal Time Coordinated (UTC).

[26] Velikost podatkovnega tipa *Integer* je 4 bajte, medtem ko je *DateTime* podatkovni tip velik 8 bajtov.

Iz teh zgornjih polj smo s pomočjo vgrajenih časovnih funkcij SQL Serverja generirali izpeljana polja, kot so YearUTC, QuarterUTC, MonthUTC, DayUTC in DateUTC za namene poročanja, analiz, pregledov …

5.2 Časovne cone in zimski/poletni čas

Enoten zapis prometa v časovni coni UTC je sicer ustrezna rešitev zapisovanja prometnih podatkov v podatkovnem skladišču, vendar pa pomeni tak zapis težavo za poslovnega uporabnika. Preprosto poslovno vprašanje, ki želi pridobiti vsoto količine vseh pogodb, sklenjenih na nekem trgu v prvem četrtletju leta 2016, zahteva od poslovnega uporabnika poznavanje:

- V kateri časovni coni se sklepajo posli na tem trgu.
- Urni zamik med časovno cono trga in UTC za ustrezno nastavitev časovnih filtrov.
- Upoštevanje zimskega/poletnega časa.
- Dodatne posebnosti geografskega območja, na katerem je bila pogodba sklenjena.

Skupina GEN-I trguje na geografskem območju, ki obsega štiri časovne pasove (WET, CET, EET in GET), tako da je vsakokratno nastavljanje časovnih filtrov glede na UTC čas za vsak trg posebej zamudno opravilo, podvrženo človeškim napakam. Zaradi tega smo se odločili, da v dimenzijo *Time* vključimo vse potrebne časovne cone (WET, CET, EET in GET), z upoštevanjem premika ure iz zimskega v poletni čas ter obratno.

Pri vpeljavi premika ure pa smo naleteli na dve posebnosti:

- Časovna cona GET nima poletnega časa.
- Turčija, ki načeloma spada v časovno cono EET, se arbitrarno odloča, kdaj bo prešla v zimski ali poletni čas. Tako je v preteklosti zaradi političnih razlogov že zamikala datum prehoda na zimski čas. Nazadnje se je to zgodilo v tem letu – 2015, ko so prehod na zimski čas zaradi volitev zamaknili za en teden. V dimenziji *Time* smo tako morali generirati novo časovno cono samo za Turčijo, s katero obvladujemo takšne primere.

Vsem dodanim časovnim conam smo generirali enaka izpeljana polja, ko smo jih generirali pri časovni coni UTC.

Tabela 6 prikazuje nekatera (ne vsa polja) časovne dimenzije *Time*. Vrednosti atributov so predstavljene na primeru datuma 1. 10. 2015 v časovni coni CET. Predstavljene so časovne cone UTC, WET, CET in EET.

Tabela 6

Naziv polja	Podatkovni tip v relacijski bazi	Primer vrednosti
DateTimeID	Smalldatetime	2015-09-30 22:00:00
DateHourStartUTC	Smalldatetime	2015-09-30 22:00:00
DateHourEndUTC	Smalldatetime	2015-09-30 23:00:00
DateUTC	Date	2015-09-30
YearUTC	SmallInt	2015
QuarterUTC	tinyint	3
MonthUTC	tinyint	9
DayUTC	tinyint	30
DateHourStartWET	Smalldatetime	2015-09-30 23:00:00
DateHourEndWET	Smalldatetime	2015-10-01 00:00:00
DateWET	Date	2015-09-30
YearWET	SmallInt	2015
QuarterWET	tinyint	3
MonthWET	tinyint	9
DayWET	tinyint	30
DateHourStartCET	Smalldatetime	2015-10-01 00:00:00
DateHourEndCET	Smalldatetime	2015-10-01 01:00:00
DateCET	Date	2015-10-01
YearCET	SmallInt	2015
QuarterCET	tinyint	4
MonthCET	tinyint	10
DayCET	tinyint	1
DateHourStartEET	Smalldatetime	2015-10-01 01:00:00
DateHourEndEET	Smalldatetime	2015-10-01 02:00:00
DateEET	Date	2015-10-01
YearEET	SmallInt	2015
QuarterEET	tinyint	4
MonthEET	tinyint	10
DayEET	tinyint	1

5.3 Poslovni dan in dimenzija *Date*

Z vpeljavo časovne dimenzije *Time* smo pokrili vsa poslovna vprašanja uporabnikov, ki vsebujejo časovno komponento. Vendar je uporaba samo časovne dimenzije pri nekaterih poslovnih vprašanjih nerodna, saj je za pravilen odgovor od uporabnika zahtevala nekaj »telovadbe«. Nerodnost bom predstavil s pomočjo poslovnega vprašanja, ki želi pridobiti količino prodane električne energije po mesecih na romunskem trgu električne energije.

Za romunski trg je značilno, da trgovanje na notranjem trgu poteka v časovni coni EET, medtem ko čezmejno trgovanje z Madžarsko in Srbijo poteka v časovni coni CET. Če bi

uporabnik za pridobitev količine uporabil atribut MonthCET dimenzije *Time*, bi dobil nepravilen rezultat za pogodbe, ki so sklenjene v časovni coni EET. Če pa bi uporabil atribut MonthEET, bi dobil nepravilen rezultat za pogodbe, ki so sklenjene v CET-u. Uporabnik mora tako za ustrezen rezultat narediti dve poizvedbi, in sicer za vsako časovno cono posebej.

Za razrešitev zgornje težave smo zato uvedli novo časovno dimenzijo *Date*, katere časovna ločljivost je en dan. Pri tej dimenziji nimamo več časovnih con in tudi ne več poletnega/zimskega časa. Ta dimenzija nam predstavlja poslovni dan pogodbe.

Čeprav se časovni dimenziji *Time* in *Date* zdita enaki, pa je ena bistvena razlika med njima, in sicer:

- Dimenzija *Time* predstavlja koledarski čas.
- Dimenzija *Date* predstavlja poslovni dan.

Ta razlika pri električni energiji ni tako opazna, ker sta koledarski čas in poslovni dan sinhronizirana, pri nekaterih drugih dobrinah pa to ne velja. Na primer pri plinu, kjer je poslovni dan definiran od 06.00 zjutraj do vključno 05.00 naslednji dan.[27]

Tako kot časovna dimenzija *Time* tudi datumska dimenzija *Date* nima številčnega nadomestnega ključa, ampak se za ključ dimenzije uporablja kar datumsko polje Date. Tudi to dimenzijo smo napolnili z dodatnimi izpeljanimi polji za potrebe poročanja in analiz.

Tabela 7 predstavlja nekaj (ne vseh) atributov dimenzije *Date*. Vrednosti atributov so vzete na primeru datuma 27. 11. 2015.

[27] Če bi uporabnik imel v podatkovnem skladišču tudi plinske pogodbe, bi bila neposredna primerjava med plinskimi in električnimi pogodbami brez uporabe dimenzije *Date* zelo zapletena.

Tabela 7

Naziv polja	Podatkovni tip v relacijski bazi	Primer vrednosti
Date	SmallDate	2015-11-27
Year	Int	2015
QuarterNumber	Tinyint	4
QuarterLabel	Varchar(5)	Q4-15
Month	Tinyint	11
MonthLabel	Varchar(6)	Nov-15
MonthName	Varchar(9)	November
Day	Tinyint	27
DayName	Varchar(8)	Friday
WeekDay	Tinyint	5
IsWeekend	Varchar(3)	No

Pri tem je treba opozoriti, da dimenzija *Date* ne vsebuje dela prostih dni, praznikov in drugih posebnih dni. Le-ti so shranjeni v drugih dimenzijah, katerih obravnava presega to diplomsko delo.

5.4 Datum posnetka stanja oziroma dimenzija SnapshotDate

Kot bomo spoznali v poglavju 6, v podatkovnem skladišču zaradi velike količine podatkov ne shranjujemo vseh izdelanih posnetkov stanja izvornih sistemov. Vsako področje ima določeno lastno politiko shranjevanja posnetkov, zato je potrebna posebna datumska dimenzija *SnapshotDate*, ki uporabniku daje informacijo o obstoju podatkov za vsako področje posebej.

Dimenzija, poleg informacije o obstoju posnetka posameznega področja, hrani tudi informacijo o zadnjem (trenutnem) posnetku v podatkovnem skladišču, kar nam pride prav v primerih, ko želimo izdelati statično poročilo z najnovejšimi podatki. Gre za polje *Last*, na katero se lahko nato sklicujemo v poizvedbah tega poročila in si s tem vedno zagotovimo najnovejše podatke.

Sama dimenzija je v primerjavi z dimenzijo *Date* in *Time* relativno preprosta, saj je sestavljena iz datumskega polja *SnapshotDate*, ki je seveda ključ dimenzije, ter bitnih polj, ki nosijo informacije o obstoju posnetka področja v podatkovnem skladišču ter zadnjem posnetku.

Tabela 8 predstavlja nekaj (ne vseh) atributov dimenzije *SnapshotDate*. Vrednosti atributov so vzete na primeru datuma 31. 5. 2015.

Tabela 8

Naziv polja	Podatkovni tip v relacijski bazi	Primer vrednosti
SnapshotDate	SmallDate	2015-05-31
ContractTrading	Bit	1
ContractSales	Bit	0
PriceCurve	Bit	1
ExchangeRate	Bit	1
Last	Bit	0

Iz tabele 8 lahko razberemo, da podatkovno skladišče na datum posnetka 31. 5. 2015 vsebuje podatke v tabeli dejstev *FactPriceCurve*Cenovne krivulje, v tabeli dejstev *FactExchangeRate* ter trgovalne pogodbe v tabeli dejstev *FactContract*.

6 Obravnava velike količine podatkov na relacijskem nivoju

Kot smo definirali v poglavju 4.4.2, imamo pri tabeli dejstev *FactContract* in *FactPriceCurve* urno granulacijo. Že takoj na začetku načrtovanja podatkovnega skladišča smo se zavedali, da bomo morali sklepati kompromise pri shranjevanju podatkov, saj zagotavljanje celotnega obsega poslovanja v vsakem posnetku stanja ni možno.

V nadaljevanju si bomo pogledali, o kolikšni količini podatkov, tako v izvornih sistemih kot v končni bazi DW_Star, sploh govorimo.

6.1 Količina podatkov na urnem nivoju v izvornih sistemih

6.1.1 Pogodbe – Trgovanje

Kot smo videli v poglavju 4.1, Skupina GEN-I trguje na skoraj vseh trgih električne energije v Evropi ter v Turčiji, Ukrajini in Gruziji. Na teh trgih Skupina sodeluje na terminskem, promptnem, izravnalnem trgu in na trgu s čezmejnimi prenosnimi zmogljivostmi. To pomeni, da imamo lahko v sistemu dnevne, večdnevne, mesečne, četrtletne, letne in večletne pogodbe. V primeru sklenjene dnevne pogodbe imamo tako v sistemu 24 zapisov, v primeru mesečne pogodbe med 720 in 744 zapisov, v primeru letne pogodbe pa imamo 8760 zapisov (v prestopnem letu 8784). Pri tem seveda lahko Skupina sklepa tudi večletne pogodbe.

Vse trgovalne pogodbe so zaprtega tipa,[28] imajo definiran začetek in konec dobave ter vsebujejo natanko en par »prevzemno mesto – dostavno mesto«. Če zanemarimo izvor energije (= prevzemno mesto), lahko povzamemo, da ima vsaka trgovalna pogodba natančno eno dostavno mesto oziroma, da bomo usklajeni s Prodajo, merilno mesto.

Na Trgovanju tako lahko za določanje obsega podatkov vzamemo število pogodb. V letu 2012 je bilo sklenjenih približno deset tisoč pogodb, v naslednjem letu 15 tisoč pogodb, v letu 2014 20 tisoč, do sredine leta 2015 pa že več kot 30 tisoč pogodb.

Če predpostavimo, da imamo v tej množici samo letne pogodbe (v realnosti imamo seveda mešanico dnevnih, mesečnih, četrtletnih, letnih in večletnih pogodb), dobimo približno 650 milijonov vrstic, ki bi jih morali vsak dan prenesti v podatkovno skladišče, oziroma približno 20 GB podatkov.

[28] Glej poglavje 2.4.

6.1.2 Pogodbe – Prodaja

Skupina GEN-I prodaja električno energijo končnim odjemalcem v Sloveniji, Italiji, Avstriji, Srbiji, Makedoniji in na Hrvaškem.

Prodajna pogodba ima v primerjavi s trgovalno pogodbo nekaj posebnosti:

- Je lahko zaprtega ali odprtega tipa.
- Obdobje dobave na pogodbi je lahko časovno omejeno ali pa je neomejeno (npr.: pogodbe odjemalcev v *skupini iv.*[29] – gospodinjstva in mali poslovni odjemalci).
- Lahko vsebuje več merilnih mest.

Pri določanju obsega podatkov na Prodaji moramo torej upoštevati merilna mesta, in ne več število pogodb. Vseh merilnih mest je v prodajnih aplikacijah[30] približno 200.000, od tega jih dobra polovica odpade na pogodbe z gospodinjstvi in malimi poslovnimi odjemalci. Ostala polovica pripada pogodbam odjemalcev v skupinah *i.–iii.*

Kot smo navedli zgoraj, pogodbe z odjemalci v *skupini iv* nimajo definiranega zaključka dobave, zato smo tem pogodbam določili štiriletno obdobje dobave. Pri pogodbah z odjemalci v skupinah *i.–iii*. pa moramo v celoti prevzeti takšno obdobje, kot je definirano v pogodbi. Obdobje dobave je lahko večletno, vendar smo za to skupino pogodb privzeli letno dobavo.

Preprost izračun zgornjih predpostavk nam vrne približno 4,4 milijarde zapisov oziroma približno 135 GB podatkov, ki bi jih morali za potrebe Prodaje vsak dan prenesti iz izvornih sistemov in obdelati v podatkovnem skladišču.

Skupaj s Trgovanjem bi tako za dnevni prenos podatkov iz izvornih sistemov v bazo DW_Source podatkovnega skladišča potrebovali 155 GB diskovnega prostora.

6.1.3 Cenovne krivulje

Cenovne krivulje so v podatkovnem skladišču shranjene na urni granulaciji. Vsaka cenovna krivulja je sestavljena iz dveh časovnih krivulj:

- **Realizacije**: gre za urni vektor že realiziranih cen na trgu z električno energijo.
- **Napovedi**: gre za napoved gibanja cen v prihodnosti.

[29] Glej poglavje 2.1 razdelek b).

[30] Glej Sliko 11 v poglavju 4.3.

Realizacijo in napoved nato združimo v enotno časovno vrsto, ki nam predstavlja cenovno krivuljo.

V nasprotju s pogodbami pri cenovnih krivuljah shranjujemo celotno obdobje, od najstarejše časovne značke realizacije v izvornem sistemu do konca časovnih intervalov napovedi. Začetek obdobja cenovne krivulje se tako lahko začne že 1. 1. 2006, konča pa leta 2020 ali več, odvisno, do katerega datuma se izdeluje napoved.

Čeprav gre tukaj, v primerjavi s pogodbami, za precej daljše obdobje, pa je obseg podatkov, v primerjavi z obsegom podatkov pogodb, precej manjši, saj imamo število cenovnih krivulj v rangu od 100 do 200.

Pri določanju obsega podatkov cenovnih krivulj smo tako privzeli 200 desetletnih cenovnih krivulj, kar nam vrne približno 17,5 milijona zapisov oziroma 500 MB podatkov, ki jih moramo dnevno prenesti iz izvornih sistemov.

6.1.4 Tečajnice

Kot smo navedli poglavju 4.4.2 pri obravnavi časovne granulacije tabele dejstev *FactExchangeRate*, tečajnice shranjujemo na dnevni granulaciji. Tako kot cenovne krivulje, so tudi tečajnice sestavljene iz:

- **Realizacije**: realiziran tečaj.
- **Napovedi**: napoved gibanja tečaja v prihodnosti.

Tako kot pri cenovnih krivuljah imamo tudi pri tečajnicah obdobje dobave definirano od najstarejšega datuma realizacije v izvornem sistemu do najmlajšega datuma napovedi. Število tečajnic je v rangu 50–100.

Za naše potrebe smo vzeli desetletni obseg podatkov, kar nam vrne 365 tisoč vrstic oziroma približno 10,5 MB podatkov.

6.2 Omejitev prometa v posnetku stanja za Pogodbe

Za zadovoljitev poslovnih zahtev uporabnikov v resnici ne potrebujemo celotnega obsega prometa pogodb v izvornih sistemih v vsakem posnetku stanja. Uporabnika načeloma bolj malo zanima stanje Trgovalnega ali Prodajnega portfelja v letu 2008, veliko bolj ga zanima trenutno stanje ter stanje portfelja v prihodnosti.

Ena prvih optimizacij za zmanjševanje količine podatkov pri tabeli dejstev *FactContract* je tako bila omejitev prometa, ki se prenaša v podatkovno skladišče, in sicer:

- Do nekega datuma v tekočem letu (npr.: do 1. marca) v podatkovno skladišče prenašamo promet pogodb, ki pade v obdobje dobave od 1. 1. preteklega leta naprej.
- Od nekega datuma v tekočem letu (npr.: od 1. marca naprej) v podatkovno skladišče prenašamo promet pogodb, ki pade v obdobje dobave od 1. 1. tekočega leta naprej.

Razlog za takšno optimizacijo je relativno preprost. V izvornih sistemih se lahko pri vnosu pogodbe zgodijo napake (npr.: napačna količina, cena, napačna krivulja za vrednotenje …), ki jih uporabniki v začetnih mesecih tekočega leta popravljajo.

Sila nerodno bi bilo, da se zaradi pravila o zajemu prometa v izvornih sistemih po odpravi napake v izvornem sistemu napaka ne bi odpravila tudi v podatkovnem skladišču. Zato smo se odločili za neko prehodno obdobje, v katerem imajo uporabniki čas za odpravo napak na pogodbah in na virih.

Z implementacijo zgornjega pravila smo tako število zapisov pri pogodbah zmanjšali s 5 milijard na približno 310 milijonov, kar je približno 9,5 GB dnevnih podatkov.

Pri tabeli dejstev *FactPriceCurve* in *FactExchangeRate* podobnega pravila zaradi poslovnih zahtev uporabnikov nismo smeli uporabiti. Pri teh dveh tabelah smo tako bili primorani hraniti ves promet za vse posnetke stanja.

6.2.1 Količina podatkov v končni bazi DW_Star

V prejšnjem razdelku smo obdelali količino podatkov, ki jih pridobivamo iz izvornih sistemov, v tem razdelku pa bomo obdelali količino podatkov, ki nastanejo pri generiranju posnetka stanja v končni bazi DW_Star.

Kot smo izvedeli v poglavju 3.4., je značilnost dimenzijskih podatkovnih skladišč denormalizacija podatkovne baze, kar pomeni, da potrebujemo več prostora za shranjevanje podatkov kot pa v normaliziranem transakcijskem sistemu. Prav tako med ETL-procesom

podatke bogatimo z dodatnimi podatki, ki so uporabnikom v pomoč pri analizah in poročilih. Če vzamemo enak obseg prometa v izvornih sistemih in ga primerjamo z istim obsegom v dimenzijskem podatkovnem skladišču, lahko pričakujemo večjo količino podatkov v dimenzijskem podatkovnem skladišču.[31]

Oglejmo si torej zdaj velikost enega generiranega posnetka stanja v bazi DW_Star za posamezno tabelo dejstev:

- Tabela dejstev *FactContract*: približno 160 GB.
- Tabela dejstev *FactPriceCurve*: približno 1.2 GB.
- Tabela dejstev *FactExchangeRate*: približno 20 MB.

Pri zgornjih številkah smo pri *FactContract* že upoštevali pravilo o omejitvi prometa v posameznem posnetku stanja.

6.2.2 Stiskanje podatkov v tabelah dejstev

Kot ena izmed rešitev za zmanjšanje količine podatkov v dnevnih posnetkih stanja se je pokazalo stiskanje podatkov v tabelah dejstev. Kot smo zapisali v poglavju 4.2, smo podatkovno skladišče zgradili na platformi SQL Serverja 2012 Enterprise Edition, ki med drugim ponuja stiskanje podatkov (angl. *Data Compression*) znotraj podatkovne baze. Na voljo smo imeli dva načina stiskanja podatkov:

- Stiskanje po vrsticah (angl. *Row Compression*).
- Stiskanje po straneh (angl. *Page Compression*), ki vsebuje tudi stiskanje po vrsticah.

Samo stiskanje podatkov ima poleg zmanjšanja velikosti podatkovne baze tudi blagodejen učinek na hitrost izvajanja poizvedb. Stisnjeni podatki so namreč shranjeni v manj podatkovnih straneh na disku, kar pomeni manj branja z diska in posledično boljše zmogljivosti. Slabost stiskanja podatkov pa je potreba po več procesorske moči za strežnik, saj moramo podatke pred izmenjavo z aplikacijami razširiti in jih med zapisovanjem nazaj v podatkovno bazo ponovno stisniti.

Smernice, kdaj, na katerih tabelah, kateri način stiskanja in njihov vpliv na hitrost izvajanja poizvedb presegajo namen diplomske naloge, zato bomo v našem primeru navedli samo našo

[31] Kot primer: v normaliziranem transakcijskem sistemu imamo vrednosti atributa ContractType, dimenzije *Contrac*t, zavedene kot numerične vrednosti od 0 naprej, medtem ko so vrednosti istega atributa v podatkovnem skladišču opisne (Phyisical, Financial, Capacity …).V prvem primeru imamo podatkovni tip Tinyint, v drugem primeru podatkovni tip Varchar (10). V podatkovnem skladišču imamo na voljo tudi več mer kot pa v izvornih sistemih, kar doprinese k večji velikosti tabele dejstev.

odločitev, da uporabimo stiskanje podatkov po straneh na vseh treh tabelah dejstev (Pogodbe, Cenovne krivulje in Tečajnice).

Po vklopu stiskanja podatkov smo dobili naslednje velikosti tabele dejstev (dnevni posnetek):

- Tabela dejstev *FactContract*: približno 40 GB.
- Tabela dejstev *FactPriceCurve*: približno 300 MB.
- Tabela dejstev *FactExchangeRate*: približno 5 MB.

Vklop stiskanja podatkov nam je zmanjšal velikost tabel dejstev za 75 %, kar kaže na to, da so naši podatki zelo »stisljivi«.

6.2.3 Politika shranjevanja posnetkov stanja za tabelo dejstev Pogodbe

Kot smo navedli v prejšnjem razdelku, v podatkovnem skladišču na dan pridelamo za približno 40 GB podatkov za tabelo dejstev *FactContract*. S politiko shranjevanje vsakega dnevnega posnetka stanja v podatkovno skladišče bi tako za enoletno hrambo posnetkov potrebovali 14,6 TB diskovnega prostora. Stroški vzpostavitve takšnega diskovnega prostora pa niso bili upravičeni (smiselni) ne iz ekonomskega, kot tudi ne iz poslovnega razloga.

Z upoštevanjem poslovnih zahtev uporabnikov smo zato določili novo politiko shranjevanja posnetkov, po kateri:

- Se vzdržuje okno zadnjih 10 posnetkov stanja v tabeli dejstev.
- Poslovnim uporabnikom se znotraj tekočega leta omogoči shranjevanje »pomembnih« posnetkov v tabeli dejstev.
- V tabeli dejstev se za pretekla leta shranjujeta dva posnetka stanja (dva za vsako leto):
    - Posnetek stanja na zadnji dan v letu.
    - Posnetek stanja na prvi dan v marcu za zagotovitev pravilnosti prometa iz preteklih let.

Pri tabeli dejstev *FactPriceCurve* in *FactExchangeRate* imamo drugačno politiko shranjevanja posnetkov, in sicer shranjujemo vse posnetke stanja.

Med pisanjem diplomskega dela smo tako imeli:

- Tabela dejstev *FactContract* je vsebovala 50 posnetkov stanja in je bila velika približno 1,98 TB.[32]

[32] Navajam samo, koliko zasedejo podatki v tabeli brez upoštevanja velikosti indeksa.

- Tabela dejstev *FactPriceCurve* je vsebovala 3100 posnetkov in je bila velika približno 221 GB.
- Tabela dejstev *FactExchangeRate* je vsebovala 1210 posnetkov stanja in je bila veliko približno 2,4 GB.

6.3 Problematika vzdrževanja podatkovnega skladišča

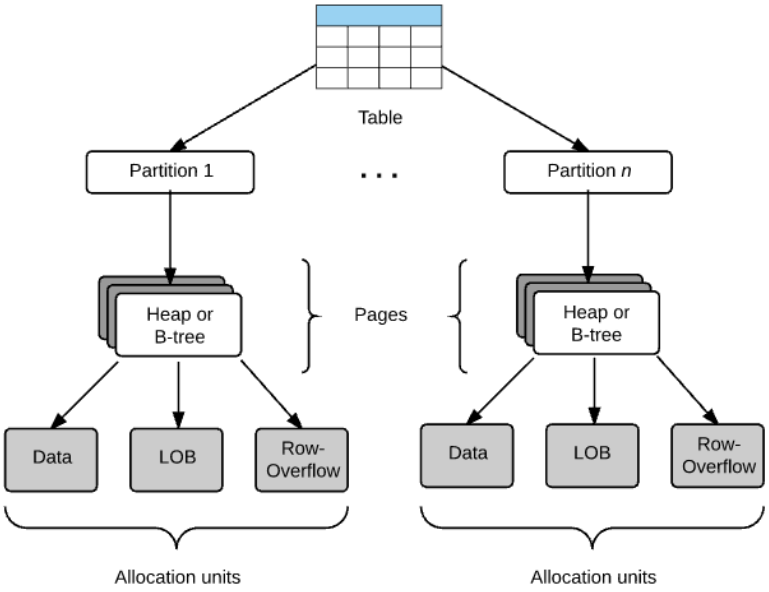
V tem razdelku bomo obravnavali problematiko vzdrževanja podatkovnega skladišča. Z vzdrževanjem mislimo predvsem na dnevno dodajanje posnetkov v podatkovno skladišče, odstranjevanje napačnih in/ali že pretečenih posnetkov in na indeksacijo podatkovnega skladišča. Pri tem je iskanje ravnotežja med hitrim polnjenjem podatkovnega skladišča in počasnimi poizvedbami ter med hitrimi poizvedbami in počasnim polnjenjem podobno plovbi med Scilo in Karibdo. V prvem primeru imamo v podatkovnem skladišču implementiranih majhno število indeksov, v drugem primeru pa veliko indeksov.

Pri iskanju ustrezne rešitve smo morali upoštevati še naslednje predpostavke:

- Osveževanje podatkovnega skladišča bo potekalo ponoči.
- Pomembna je hitrost poizvedb, hitrost polnjenja je v ozadju, vendar pa se mora polnjenje podatkovnega skladišča zaključiti pred 6.00 zjutraj.
- Zaradi politike hranjenja podatkov v tabeli dejstev *FactContract* bomo imeli tako dodajanje kot brisanje vrstic v tabelah dejstev v nočnem ETL-opravilu.
- Tipičnih poizvedb uporabnikov na tabeli dejstev *FactContract*, na katerih bi lahko temeljili optimizacijo indeksacije, med implementacijo podatkovnega skladišča ne bomo imeli na voljo.

6.3.1 Particioniranje

Za lažje razumevanje particioniranja tabel si bomo najprej ogledali, kako je tabela sploh organizirana v SQL Serverju 2012 ter se spoznali z gradniki in koncepti particioniranja.



Slika 16: Organizacija tabele [5]

Tabela je v SQL Serverju 2012 zbirka podatkovnih strani (angl. *pages*), v katerih so zapisane vrstice podatkov. Vsaka podatkovna stran je velika 8 KB. Ko kreiramo tabelo, je le-ta privzeto vsebovana na eni particiji.

Če ima tabela definiran povezani indeks, se podatkovne strani tabele organizirajo v podatkovno strukturo B-drevesa (angl. *B-tree*), sicer so v podatkovni strukturi kopice (angl. *heap*). Kadar ima tabela več particij, ima vsaka particija lastno B-drevo ali kopico. Katera enota (angl. *allocation unit*) nato skrbi za upravljanje podatkovnih strani, je odvisno od tipa/vrste stolpcev, ki so na teh straneh:

- **Data**: se uporablja za upravljanje vseh podatkovnih vrstic, razen če gre za pri tem za podatkovne vrstice LOB.
- **LOB (angl. *Large OBject*)**: se uporablja za upravljanje vseh podatkovnih vrstic tipa *text*, *ntext*, *image*, *xml*, *varchar(max)*, *nvarchar(max)*, *varbinary(maxy)* ali *CLR* (*Common Language Runtime*) uporabniško definiranih podatkovnih tipov.
- **Row-Overflow**: se uporablja za upravljanje vseh podatkovnih vrstic variabilnih podatkovnih tipov (varchar, nvarchar, varbinary ali sql_vairant), ki presegajo 8 KB omejitev velikosti vrstice.

Particija (angl. *partition*): je uporabniško definirana enota, ki hrani podatkovne strani tabele. Če tako tabelo particioniramo po nekem stolpcu in pogoju, se njene podatkovne vrstice v tabeli horizontalno razbijejo in umestijo v podatkovne strani ustrezne particije. Čeprav je tabela na fizičnem nivoju ločena, se na logičnem nivoju obnaša kot enotna entiteta, ko se nad

njo izvajajo poizvedbe. Particije tabele so lahko vsebovane v več kot eni datotečni skupini (angl. *filegroup*) podatkovne baze. V SQL Serverju 2012 smo omejeni na 15.000 particij.

Particijska funkcija (angl. *Partition function*): je objekt v podatkovni bazi, ki podatkovne vrstice tabele razvršča v ustrezno particijo iz množice particij glede na vrednost stolpca, po katerem particioniramo tabelo. Stolpec, po katerem particioniramo tabelo, se imenuje particijski stolpec (angl. *Partitioning column*). Od particijske funkcije je odvisno število particij, ki jih ima tabela in kako so definirane meje vsake posamezne particije.

Particijski stolpec (angl. *Partitioning column*): je stolpec tabele, po katerem particioniramo tabelo.

Particijska shema (angl. *Partition scheme*): je objekt v podatkovni bazi, ki vsebuje navezavo particij tabel z datotečnimi skupinami podatkovne baze. Glavni razlog, zakaj bi particije neke tabele razdelili na več datotečnih skupin, je varnostno kopiranje. Varnostno kopiranje lahko poteka neodvisno po posameznih datotečnih skupinah.

Poravnani indeks (angl. *Aligned index*): Gre za indeks, ki je vsebovan v isti particijski shemi, kot je tabela, na kateri je indeks definiran. Kadar sta tabela in indeks poravnana, SQL Server hitro (v trajanju nekaj sekund) izvaja particijsko preklapljanje (angl. *partition switching*).

Particijsko izločanje (angl. *Partition elimination*): Gre za proces optimizacije poizvedb, ki omogoča, da dostopamo samo do relevantnih particij, ki izpolnjujejo merila filtrov v poizvedbi.

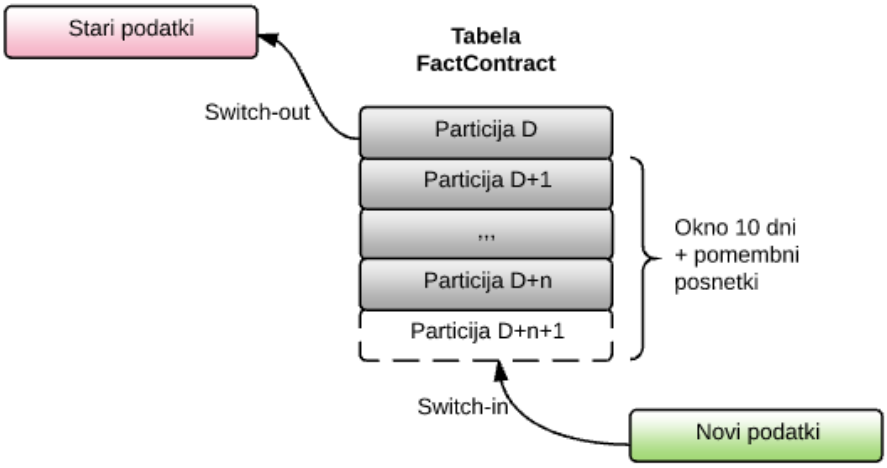
Particijsko preklapljanje (angl. *Partition switching*): Gre za proces, pri katerem SQL Server jemlje particije ven iz tabele (ang, *switch-out*) in jih nato ponovno vrača (angl. *switch-in*) v tabelo. Tabela v tem vmesnem času ne vidi podatkov v tej particiji.

Prednosti particioniranja velikih tabel bi tako lahko združili v naslednje točke:

- Izvajanje vzdrževalnih operacij nad eno ali več particijami tabele poteka hitreje, saj se izvajajo samo nad podatki, ki so vsebovani v teh particijah, in ne nad celotnimi podatki tabele. Primer take operacije je osvežitev indeksa, ki poteka samo na izbranih particijah, in ne na celotni tabeli.
- Branje podatkov poteka hitreje, saj s pomočjo particijskega izločanja izvajamo poizvedbe samo nad podatki, ki so vsebovani v izbranih particijah.
- Brisanje podatkov je izjemno hitro v primeru, če brišemo podatke z enakimi pogoji, kot smo jih uporabili pri particioniranju tabele. Operacija, ki lahko v velikih tabelah traja celo večnost, v tem primeru traja nekaj sekund.

6.3.1.1 Implementacija particioniranja

Oglejmo si, kako smo particioniranje uporabili v našem primeru. Kot smo navedli v poglavju 6.2.3, imamo vzpostavljeno politiko shranjevanja posnetkov stanja za tabelo dejstev *FactContract*. Le-ta zahteva desetdnevno okno posnetkov stanja plus shranjevanje uporabniško »pomembnih« posnetkov stanja. Zahtevo uporabnikov lahko vizualiziramo s sliko 17.



Slika 17: Politika shranjevanja podatkov v tabeli dejstev Pogodbe

Da smo tabelo dejstev *FactContract* lahko particionirali, smo morali izvesti naslednje korake (po vrsti):

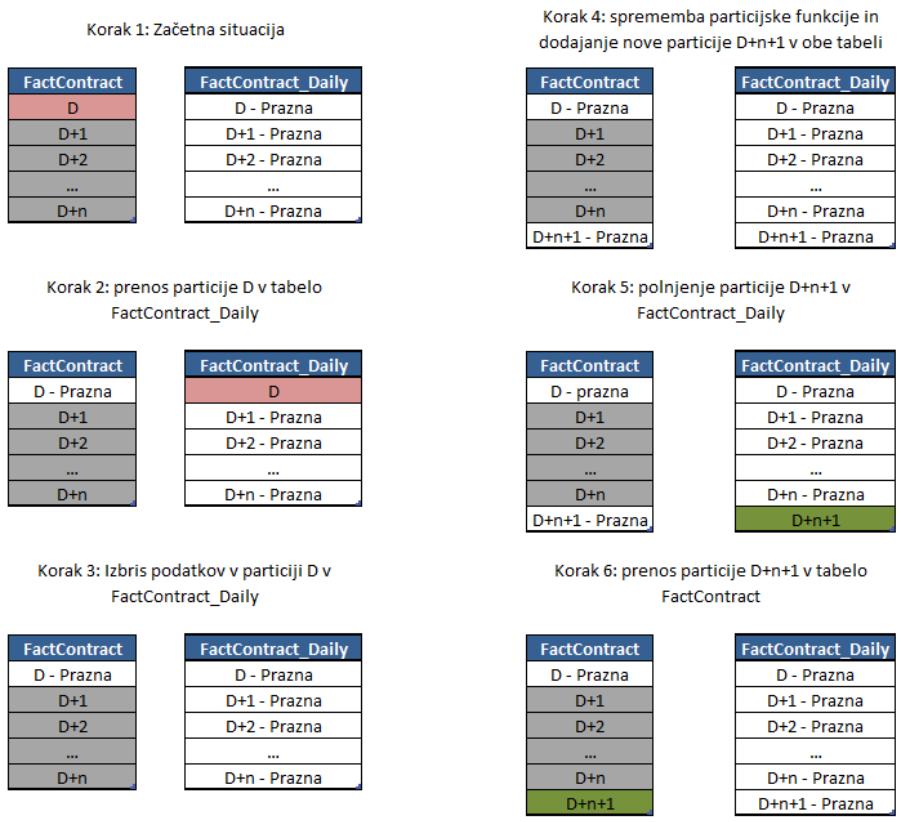
- **Kreiranje dodatnih datotečnih skupin**: V našem primeru smo ta korak preskočili, saj smo se zaradi lažje implementacije in vzdrževanja particijske funkcije omejili na eno datotečno skupino.
- **Kreiranje particijske funkcije**: Glede na politiko shranjevanja smo za particijski stolpec določili polje *SnapshotDate* ter v particijsko funkcijo ročno vpisali prvih deset zaporednih datumov za implementacijo desetdnevnega drsečega okna.
- **Kreiranje particijske sheme**: Glede na to, da smo izvedli particioniranje tabel na enotni datotečni skupini je particijska shema enostavna, kjer so vse particije vsebovane v eni datotečni skupini.
- **Kreiranje tabele dejstev Pogodbe**: Na koncu smo z uporabo particijske sheme in particijskega stolpca kreirali particionirano tabelo dejstev *FactContract*.

6.3.1.2 Implementacija drsečega okna

Koncept drsečega okna je vzdrževati enako število particij v particionirani tabeli. Ko med nočnim osveževanjem tabela Pogodbe generiramo nov posnetek, v particionirani tabeli generiramo novo particijo, v katero shranimo nove podatke, hkrati pa iz tabele odstranimo, glede na particijsko funkcijo, particijo s starimi podatki. V našem primeru pa je bil koncept drsečega okna malo spremenjen, saj smo morali vzdrževati desetdnevno okno z dodatnimi, uporabniško pomembnimi posnetki. To pomeni, da imamo skozi čas v tabeli vedno več particij, in ne več samo deset, kot smo jih generirali na začetku.

Sama realizacija prilagojenega drsečega okna je predvidevala naslednje korake, kot so razvidni na sliki 18:

1. Kreiranje in uporabo dodatne tabele FactContract_Daily, ki ima popolnoma enako strukturo (vključno s particijami in indeksi) kot tabela dejstev *FactContract*. Pred particijskim preklapljanjem smo morali poskrbeti, da so bile particije dodatne tabele FactContract_Daily prazne.
2. Nato smo prenesli ustrezno particijo s pomočjo particijskega preklapljanja iz tabele *FactContract* v FactContract_Daily.
3. Izbrisali smo podatke v preneseni particiji v tabeli FactContract_Daily.
4. S pomočjo particijskega preklapljanja smo nato prazno particijo prenesli nazaj v tabelo *FactContract*. Nato smo spremenili particijsko funkcijo, v katero smo dodali datum novega posnetka, ter hkrati generirali novo »prazno« particijo v tabeli *FactContract* in FactContract_Daily. S pomočjo particijskega preklapljanja smo nato to novo particijo prenesli v FactContract_Daily.
5. Napolnili smo tabelo FactContract_Daily z novimi podatki.
6. S pomočjo particijskega preklapljanja smo nato prenesli particijo z novimi podatki nazaj v tabelo *FactContract*.



Slika 18: Polnjenje in praznjenje tabele dejstev Pogodbe s pomočjo pomožne tabele in particioniranja

Slabost takšnega polnjenja in praznjenja je množica praznih particij v tabeli dejstev *FactContract*. Glede na to, da imamo v SQL Serverju 2012 na voljo 15 tisoč particij, imamo, ob trenutni definiciji particijske funkcije, zaloge še za približno 40 let, kar zadostuje našim potrebam.

Tabeli dejstev *FactPriceCurve* in *FactExchangeRate* smo prav tako particionirali na enak način kot *FactContract*, vendar pa pri njiju ne izvajamo korakov 1 3, ampak samo dodajamo nove particije v tabeli – korake 4–6 na sliki 18.

6.3.2 Indeksacija v podatkovnem skladišču

Težavo indeksacije tabel v podatkovnem skladišču smo reševali z indeksom ColumnStore. V tem razdelku si bomo najprej pogledali, kaj sploh je indeks tabele, kateri indeksi so v SQL Serverju najpogostejši, ter definicijo in uporabo indeksa ColumnStore v praksi.

6.3.2.1 Indeks

Indeks je podatkovna struktura na disku, vezana na tabelo ali pogled, ki omogoča hitrejše vračanje vrstic poizvedbe iz tabele ali pogleda. Indeks vsebuje ključe, ki so zgrajeni iz enega ali več stolpcev tabele ali pogleda in so v SQL Serverju shranjeni v obliki B-drevesa. Podatki znotraj indeksa se shranjujejo na straneh, velikih 8 KB. Sama organizacija indeksa je enaka organizaciji tabele na sliki 16.

V SQL Serverju 2012 se najpogosteje srečujemo s:

- **Povezanim indeksom (angl. *Clustered Index*)**
    - Povezani indeks podatke v tabeli ali pogledu uredi in shrani po vrstnem redu glede na vrednosti ključa. Ključ sestavljajo stolpci, ki so vključeni v definicijo indeksa. Tabela ima lahko definiran samo en povezani indeks, saj so podatki znotraj tabele lahko urejeni samo na en način.
    - Če ima tabela vzpostavljen povezani indeks, so njene podatkovne vrstice shranjene v obliki B-drevesa. Tako tabelo imenujemo tudi povezana tabela (angl. *Clustered table*).
    - Podatki so v indeksu shranjeni v vrstičnem formatu (angl. *row store format*).
- **Nepovezanim indeksom (angl. *NonClustered Index*)**
    - Tabela (tudi povezana tabela) ima lahko več nepovezanih indeksov.
    - Nepovezani indeksi imajo lastno strukturo ločeno od podatkovnih vrstic. Vsaka vrstica vsebuje ključ (angl. *key value*) in kazalnik (angl. *row locator*). Ta kazalnik nato kaže na podatkovno vrstico v povezanem indeksu (če ima tabela definiran povezani indeks) ali pa na podatkovno vrstico, ki vsebuje ključ indeksa v kopici.
    - Podatki so v indeksu shranjeni v vrstičnem formatu.

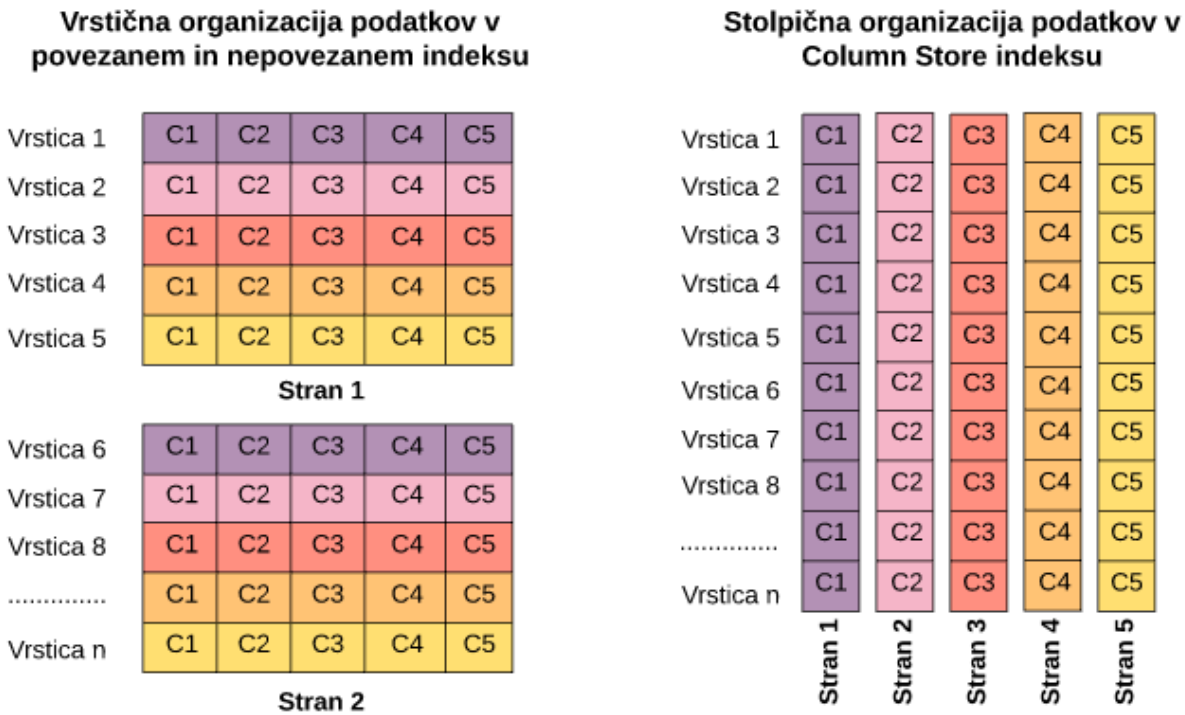
SQL Server 2012 seveda vsebuje še nekatere druge tipe indeksov (npr.: XML, Spatial, Full-text …), ki pa presegajo obseg diplomske naloge.

6.3.2.2 Nepovezani indeks ColumnStore[33]

Microsoft je v SQL Serverju 2012 predstavil novo vrsto nepovezanega indeksa – ColumnStore indeks, za katerega trdijo, da lahko prinese stokratno hitrostno pohitritev izvajanja poizvedb ter desetkratno boljšo kompresijo podatkov v indeksu v primerjavi s povezanim in nepovezanim indeksom [21].

Ti dve trditvi sta nam vzbudili pozornost, saj smo se v projektni skupini zavedali pomembnosti postavljanja pravilne strategije indeksiranja podatkovnega skladišča. Z namenom preverbe teh trditev smo postavili testno okolje, v katerem smo izvedli hitrostne teste z različnimi vrstami indeksiranja. Rezultati, ki smo jih dobili, so v veliki večini primerov potrjevali navedbe, zato si oglejmo, kako so to izboljšanje dosegli.

Prva razlika med indeksom ColumnStore in drugimi indeksi je v načinu shranjevanja podatkov. Povezani in nepovezani indeks podatke v indeksu shranjujeta v obliki B-drevesa vrstično, indeks ColumnStore podatke shranjuje stolpčno, kot je prikazano na sliki 19.



Slika 19: Vrstični in stolpčni format

[33] Značilnosti in omejitve indeksa ColumnStore veljajo za SQL Server 2012. V novejših verzijah nekaterih omejitev ni več.

Poleg stolpčne organizacije podatkov so k večji hitrosti poizvedb pripomogle še nekatere druge izboljšave:

- Izboljšana kompresija podatkov. Medtem ko so si lahko podatki v vrsticah v tabeli med seboj zelo različni (so različnega tipa), so podatki v stolpcih enakega tipa. To pomeni, da je med podatki v stolpcu lahko veliko enakih vrednosti in posledično je stiskanje podatkov učinkovitejše. Učinkovitejše stiskanje podatkov pomeni, da z enim fizičnim bralnim ciklom prenesemo več podatkov (strani) z diska v delovni spomin sistema in s tem izboljšamo verjetnost zadetka v pomnilniku. Ker imamo manj dostopov do diska, ki je v računalništvu eden najpočasnejših dostopov, imamo hitrejše izvajanje poizvedb. Z boljšo kompresijo podatkov v indeksu pridobimo tudi diskovni prostor.

- Ko izvajamo poizvedbe nad tabelo s povezanim ali nepovezanim indeksom, v delovni spomin prenesemo vse stolpce (preberemo celotno stran), čeprav teh stolpcev pozneje sploh ne potrebujemo pri poizvedbi. Pri indeksu ColumnStore pa v delovni spomin prenesemo samo stolpce, ki so potrebni za izvedbo poizvedbe. S tem in s kombinacijo izboljšanega stiskanja podatkov dosežemo manjše število bralnih dostopov do diska in posledično močno pohitrimo izvajanje poizvedb.

- Indeks ColumnStore ne pozna koncepta ključnih stolpcev, kot je znan pri povezanih in nepovezanih indeksih. Pri ColumnStore indeksu prav tako nismo omejeni na 16 ključnih stolpcev kot pri povezanih ali nepovezanih indeksih, ampak na 1024 stolpcev. Ta značilnost pa nam je v projektni skupini zelo olajšala odločanje o indeksiranju podatkovnega skladišča, saj ni bilo več potrebe po iskanju »tipičnih« uporabniških poizvedb in na njihovi podlagi načrtovanja ključnih stolpcev. Ker se v delovni spomin prenašajo samo stolpci, potrebni za izvedbo poizvedbe, smo v indeks ColumnStore dodali kar vse stolpce tabele.

Vendar pa je indeks ColumnStore prinesel tudi nekaj omejitev:

- Indeksa ColumnStore ne moremo spreminjati. V tem primeru moramo indeks uničiti in ga nato ponovno zgraditi. Če to počnemo na veliki tabeli, je lahko to kar velik problem.
- Indeks ColumnStore se ne ažurira. To pomeni, da so tabele, ki imajo definiran ta indeks, »Read-Only«. Če bi želeli v tabelo vnesti nove vrstice, bi morali najprej onemogočiti indeks, vnesti vrstice in ga nato ponovno zgraditi. Pri tabelah z veliko vstavljanja je lahko to kar velik problem. V našem primeru, kjer imamo paketno vstavljanje v sklopu nočnega opravila, je problem precej manjši.

- ColumnStore je prinesel nekaj omejitev, kateri podatkovni tipi se lahko vključijo v indeks in kateri ne. Te omejitve smo morali pri projektni skupini upoštevati pri načrtovanju podatkovne baze.

6.3.2.3 Implementacija indeksacije

Največji problem pri implementaciji indeksa ColumnStore za naše potrebe je njegova nezmožnost ažuriranja. Dnevno onemogočanje in ponovno postavljanje indeksa nad 2 TB veliko tabelo dejstev *FactContract* zaradi novih podatkov namreč traja več kot en dan, tako da smo morali poiskati rešitev v obliki particioniranja tabel in particijskega preklapljanja. Postavili smo naslednji proces:

- Tabele dejstev *FactContract*, *FactPriceCurve* in *FactExchangeRate* smo najprej particionirali in pripravili na dnevno polnjenje/praznjenje, kot je prikazano v poglavju 6.3.1.1.
- Že takoj na začetku smo na te tabele in pomožne tabele postavili indekse ColumnStore.
- V indeks ColumnStore smo vključili vse stolpce tabele.
- Pri nočnem osveževanju tabel smo najprej uničili indeks ColumnStore na particijah pomožnih tabel, nato smo te particije napolnili z novimi podatki, na njih ponovno zgradili indeks ColumnStore ter jih nato s pomočjo particijskega preklapljanja prenesli nazaj v glavne tabele.

Zgornja rešitev se zelo dobro obnese pri vsakodnevnem polnjenju podatkovnega skladišča, slabo pa se obnese pri pokrivanju povečevanja/zmanjševanja števila stolpcev tabele dejstev zaradi spremenjenih poslovnih zahtev. Žal v tem primeru ni bližnjice, ampak je treba celoten indeks nad tabelo uničiti in ga nato ponovno zgraditi, kar v našem primeru pomeni večdnevno opravilo.

Rešitev, s katero smo problem zgolj omilili z vidika končnega uporabnika (v smislu dostopa do podatkov), je bila naslednja:

- Najprej smo generirali novo tabelo in novo pomožno tabelo, ki sta že vsebovali novo strukturo glede na zahteve poslovnih uporabnikov.
- Novo tabelo smo nato particionirali.
- Iz stare tabele dejstev smo v novo pomožno tabelo v ustrezno particijo prenesli podatke iz najnovejše particije. Nova polja smo napolnili v skladu z definicijami.
- Nad particijo nove pomožne tabele smo nato zgradili indeks ColumnStore ter s pomočjo particijskega preklapljanja prenesli v novo glavno tabelo.

- Proces smo ponavljali z vsemi obstoječimi particijami v stari glavni tabeli.
- Posledično so s tem procesom uporabniki imeli najprej dostop do najnovejših podatkov v novi tabeli in šele med tednom dostop do starejših podatkov.

Indeks ColumnStore smo postavili nad vsemi tabelami v podatkovni bazi DW_Star.

6.3.2.4 Vpliv na diskovni prostor

Na koncu pa omenimo še vpliv indeksacije tabel na velikost tabele na diskovnem prostoru. Kot smo spoznali na začetku tega poglavja, je indeks dodatna (razen v primeru povezanega indeksa) podatkovna struktura na disku, vezana na tabelo, in kot taka prispeva k večji porabi diskovnega prostora. V skrajnem primeru, če v nepovezani indeks vključimo vse stolpce tabele, lahko podvojimo velikost tabele na diskovnem polju. V našem primeru, kjer smo za indeksiranje uporabili indeks ColumnStore, ki je vseboval vsa polja tabele, smo velikost tabel povečali za približno eno tretjino.

Končna poraba diskovnega prostora največjih tabel podatkovnega skladišča je tako naslednja:

- Tabela dejstev *FactContract*: 2,63 TB.
- Tabela dejstev *FactPriceCurve*: 293,93 GB.
- Tabela dejstev *FactExchangeRate*: 3,19 GB.

7 Obravnava velike količine podatkov v MOLAP-bazi

V poglavju 6. smo opisovali težave velike količine podatkov in rešitve, s katerimi smo te težave obvladali v relacijski podatkovni bazi. V tem poglavju pa si bomo ogledali, na kakšen način smo se spopadli z veliko količino podatkov v analitičnih podatkovnih bazah.

V poglavju 3.8.2 smo navedli, da potrebujemo za ažurnost MOLAP-kock njihovo redno procesiranje. Procesiranje kock je operacija, pri kateri, v odvisnosti od tipa procesiranja, podatke v MOLAP osvežimo oziroma nadomestimo z novimi podatki iz virov podatkov – v našem primeru beremo iz relacijskega podatkovnega skladišča.

Poznamo več vrst procesiranja [6]:

Process Full: Ko izvedemo Process Full nad analitičnim objektom, se podatki tega objekta razveljavijo – izbrišejo. Po razveljavitvi podatkov se objekt osveži s podatki iz relacijske podatkovne baze. Ta vrsta procesiranja je potrebna, če imamo strukturne spremembe na objektu (npr.: preimenovanje/dodajanje/odstranjevanje atributa v dimenziji). Process Full lahko izvedemo na vseh analitičnih objektih (dimenzije, kocke, skupine mer, particije skupine mer …).

Process Default: Ko izvedemo Process Default nad analitičnim objektom (dimenzija, kocka, skupine mer), le-ta najprej preveri stanje objekta in nato izvede operacije, ki so potrebne, da objekt prestavijo iz neprocesiranega stanja ali delno procesiranega stanja v procesirano stanje. Če se pri preverbi stanja objekta ugotovijo strukturne spremembe, se izvede Process Full. Process Default lahko izvedemo na vseh analitičnih objektih.

Process Clear: Ko izvedemo Process Clear nad analitičnim objektom, se podatki tega objekta razveljavijo – izbrišejo. Po razveljavitvi se objekt ne osveži z novimi podatki – ostaja prazen. Process Clear lahko izvedemo na vseh analitičnih objektih.

Process Indexes: Pred izvedbo Process Indexes nad analitičnim objektom moramo zagotoviti, da objekt vsebuje podatke, sicer se procesiranje prekine. Process Indexes ohrani podatke v objektu ter ponovno zgradi »indekse«. Če je objekt dimenzija, proces zgradi bitni indeks dimenzije, če je objekt particija skupine mer, proces zgradi agregacije in bitni indeks skupine mer. Process Indexes lahko izvedemo nad dimenzijo, kocko, skupino mer in particijo skupine mer.

Process Update: To vrsto procesiranja lahko izvedemo samo nad dimenzijo. Med izvajanjem procesiranja le-ta ugotavlja spremembe nad podatki v dimenziji in te spremembe lahko vplivajo na stanje podatkov v skupinah mer:

- Če se samo dodajajo novi ali spreminjajo obstoječi člani dimenzije, to ne vpliva na povezane podatke v particijah skupine mer.
- Če se neki član dimenzije odstrani oziroma se mu spremeni relacija do povezanih podatkov v particijah skupine mer (npr.: če se partner preseli iz Ljubljane v Maribor), se povezani podatki in povezane agregacije v particijah skupine mer odstranijo.
- To pomeni, da je treba po izvršitvi Process Update izvesti še Process Indexes nad prizadetimi particijami skupine mer, ki nam popravi agregacije ter bitne indekse skupine mer.
- Težava procesiranja Process Update je v tem, da je zaradi narave procesiranja počasnejša operacija od operacije Proces Full.

Če torej primerjamo možnosti, ki jih imamo na voljo pri procesiranju kocke, z dogajanjem pri nočnem polnjenju tabel dejstev v našem primeru ugotovimo naslednje:

- V dimenziji vedno posodabljamo obstoječe zapise ali pa dodajamo nove zapise oziroma generiramo nove zapise zaradi pravil SCD2.
- V tabele dejstev vedno ali dodajamo nove particije ali pa odstranjujemo stare particije. Ne izvajamo posodabljanja podatkov na obstoječih particijah tabele oziroma so te operacije izjema.

V našem podatkovnem skladišču imamo realizirani dve kocki – arhivsko (Cube) in trenutno (Cube_Current), kjer je edina razlika med njima v številu posnetkov stanja. Trenutna kocka vsebuje samo zadnja dva (najnovejša) posnetka vseh področij, medtem ko arhivska kocka vsebuje vse posnetke vseh področij.

Kot najustreznejša strategija se je za procesiranje trenutne kocke izkazala izbira Process Full dimenzij in skupin mer. Čeprav s to vrsto procesiranja vse podatke v trenutni kocki izbrišemo in jih nato ponovno napolnimo, je bilo trajanje operacije krajše od vseh drugih načinov procesiranja.

Pri procesiranju arhivske kocke pa izbira Process Full nad celotno kocko ni prišla v poštev, saj bi morali v tem primeru v nočnem obdobju sprocesirati približno 2,6 TB podatkov, kar pa na trenutni strojni opremi traja več dni. Za procesiranje arhivske kocke smo tako morali poiskati drugačno rešitev.

7.1.1 Zakaj trenutna kocka

Za uvedbo Trenutne kocke smo se odločili zaradi naslednjih prednosti:

- Procesiranje trenutne kocke je zelo hitro (v primerjavi s procesiranjem celotne kocke) in ga je možno izvesti znotraj nočnega osveževanja podatkovnega skladišča.
- Dodajanje novih atributov v dimenzije MOLAP-kocke povzroči procesiranje celotne kocke, z uvedbo trenutne kocke pa smo tako lažje sledili zahtevam poslovnih uporabnikov, saj smo zahtevane spremembe lahko implementirali v nekaj dnevih.
- Načeloma strukturnih razlik med trenutno kocko in arhivsko kocko ni, razen v številu posnetkov stanja. Uporabnik lahko uporablja iste analize nad trenutno kocko ali nad arhivsko kocko.
- Zaradi manjše količine podatkov je trenutna kocka bolj odzivna, kar je pripomoglo k boljšemu začetnemu sprejemu pri uporabnikih.
- V izjemnih primerih (npr.: izpad dnevnega poročanja) nam trenutna kocka omogoča hitro saniranje incidenta.
- Glede na to, da velika večina uporabnikov pregleduje podatke iz najnovejšega posnetka, nam trenutna kocka zagotavlja večjo fleksibilnost pri vzdrževanju arhivske kocke.

7.1.2 Particioniranje arhivske kocke

Za razrešitev našega problema procesiranja arhivske kocke smo uporabili, podobno kot v relacijskih tabelah dejstev, koncept particioniranja. SQL Server 2012 Enterprise Edition namreč med svojimi funkcionalnostmi ponuja tudi funkcionalnost particioniranja skupine mer v kocki analitične podatkovne baze. To nam je prineslo naslednje prednosti:

- možnost procesiranja posamezne particije skupine mer,
- možnost uporabe paralelizma pri procesiranju particij skupine mer
- ter najpomembnejšo prednost – dosegljivost kocke med procesiranjem.

Vse te možnosti smo upoštevali pri izbiri strategije procesiranja. Ker smo zdaj imeli particije v skupini mer, smo zdaj izbrali Process Default za dimenzije ter Process Full za posamezne particije skupine mer.

Tako kot pri particioniranju tabel dejstev v relacijski podatkovni bazi smo morali tudi tukaj določiti particijsko funkcijo:

- Najprej smo morali našo arhivsko kocko razbiti na particije po dnevih posnetka stanja in se na ta način izenačiti s particijsko funkcijo na relacijskih tabelah dejstev. To

izenačitev smo potrebovali zaradi politike shranjevanja posnetkov v tabeli dejstev *FactContract*.[34] Tako smo na dokaj enostaven način dobili informacijo, katere particije v kocki je treba ponovno procesirati zaradi sprememb (umik/dodajanje novih particij) v tabeli dejstev *FactContract*.

- Nato pa smo znotraj dnevne particije skupine mer področja Pogodbe v kocki le-to dodatno razbili po viru podatkov, saj smo želeli izkoristiti uporabo paralelizma pri procesiranju particij.

S tako določeno particijsko funkcijo smo sicer naleteli na težavo pri avtomatizaciji procesa procesiranja, saj orodja, s katerimi smo želeli to avtomatizirati, niso ponujala rešitev »out-of-the-box«. Zato smo morali vso logiko procesiranja particij realizirati v prilagojeni programski C# komponenti.

7.1.3 Vpliv na diskovni prostor

Za zaključek poglavja pa poglejmo še »slabo« lastnost MOLAP-kock, to je dodatna poraba diskovnega prostora. Kot smo izvedeli v poglavju 4.8.2, v MOLAP-kockah podatke shranimo v posebne večdimenzijske tabele, kar pomeni podvojene podatke. Ena kopija podatkov je v relacijski podatkovni bazi, druga kopija pa v MOLAP-kocki.

Velikost MOLAP-kock je naslednja:

- Trenutna kocka (Cube_Current): 64 GB.
- Arhivska kocka (Cube): 2.36 TB.

Za celotno podatkovno skladišče tako potrebujemo ca. 6 TB diskovnega prostora.

[34] Glej poglavje 7.3.1.

8 Sklep

V diplomskem delu smo predstavili, kako smo izvedli projekt »Izgradnje podatkovnega skladišča v Skupini GEN-I«. V njem smo osvetlili nekatere probleme, s katerimi smo se srečali med implementacijo, ter katere rešitve smo pri tem uporabili za njihovo razrešitev.

Projekt se je na koncu izkazal za veliko večji projekt, kot je bilo prvotno načrtovano. Če smo na začetku bolj ali manj pričakovali izziv na tehničnem področju – postavitev podatkovnega skladišča, pa nismo predvideli toliko težav, ki so nastale zaradi sprememb v obstoječih poslovnih procesih. Kajti, ko smo z razvojem podatkovnega skladišča prišli do izvornih sistemov, smo ugotovili, da le-ti vsebujejo veliko »okostnjakov v omari«, ki so nam, po večletnem nenamernem zanemarjanju, padli v naročje.

Zato smo morali najprej sanirati napake, popraviti poslovni proces v ozadju, ki je povzročal napake, in ponovno preveriti vsebino izvornih sistemov ter njihov vpliv na stanje Trgovalnega in Prodajnega portfelja. Bilo je neko obdobje projekta, ko smo se bolj ukvarjali z usklajevanjem definicij, terminologije, odpravljanjem napak, optimiziranjem in spreminjanjem poslovnih procesov kot pa samimi tehničnimi aspekti izgradnje podatkovnega skladišča. Seveda je zaradi tega prihajalo do internih trenj z uporabniki znotraj posameznih služb/oddelkov, ki niso bili ravno navdušeni nad tem dogajanjem. V teh primerih nam je zelo prav prišla podpora iz poslovodstva Skupine GEN-I, ki je poskrbelo za dokaj hitro razreševanje teh problemov.

Po uvedbi podatkovnega skladišča v letu 2012 smo v naslednjih letih videli strmo rast števila uporabnikov in oddelkov, ki so uporabljali njegove storitve oziroma podatke. Če so na začetku podatkovno skladišče uporabljali predvsem uporabniki za spremljanje gibanja vrednosti portfelja na Trgovanju in Prodaji, so se jim v naslednjih letih pridružili še trgovci z električno energijo, analitiki trga, uporabniki v zakladništvu, računovodstvu, uporabniki v oddelku za oceno tveganja kreditnega tveganja ter drugi.

Število uporabnikov podatkovnega skladišča se je še povečalo, ko smo mu z nadgradnjami dodajali nova področja. Vanj smo pripeljali področja Trgovanje s plinom, Trgovanje s finančnimi inštrumenti, Telemetrijo ter še nekatera druga področja. Ko se je za namene diplomskega dela preverjala uporaba podatkovnega skladišča, smo ugotovili, da je v Skupini GEN-I zelo malo služb/oddelkov, ki njegovih podatkov ne uporablja, pa še to gre predvsem za službe/oddelke, ki nimajo veliko skupnega z osnovno dejavnostjo Skupine (npr.: kadrovska služba).

Zgornja ugotovitev pa pomeni, da so uporabniki zelo dobro sprejeli podatkovno skladišče in da je le-to postalo eden izmed temeljev v delovnih procesih Skupine GEN-I, ki se bo v prihodnosti še naprej uporabljalo, razvijalo in nadgrajevalo.

9 Literatura in viri

[1] (2015) Agencija za energijo, *Udeleženci na trgu z električno energijo.* Dostopno na:

http://www.agen-rs.si/udelezenci-na-trgu-z-elektricno-energijo [Dostop 21. 9. 2015]

[2] (2010) D. Paravan, *Osnove trga z električno energijo*. Dostopno na: http://www.powerlab.uni-mb.si/Slo/download/Literatura/Paravan-Osnove_trga_z_elektricno_energijo.pdf [Dostop 14. 9. 2015]

[3] R. Kimball in M. Ross, *The Data Warehouse Toolkit (2nd Ed.)*, New York: John Wiley & Sons, Inc., 2012.

[4] Skupina GEN-I, *Letno poročilo skupine GEN-I za leto 2014*, Krško: GEN-I, d.o.o., 2015.

[5] (2015) Microsoft, *Table and Index Organization.* Dostopno na: https://technet.microsoft.com/en-us/library/ms189051(v=sql.105).aspx

[Dostop 24. 1. 2016]

[6] (2015) Microsoft, *Processing Options and Settings (Analysis Services)*. Dostopno na: https://msdn.microsoft.com/en-us/library/ms174774.aspx [Dostop 2. 2. 2016]

[7] F. P. Sioshansi, W. Pfaffenberger, *Electricity Market Reform: An International Perspective*, Oxford: Elsevier, 2006.

[8] Evropska komisija, *First Benchmarking Report on the implementation of the Internal Electricity and Gas Market*, Commission Staff WP, Brussels, 2002.

[9] T. Ravbar, *Električni trg in zakonodaja Slovenije*. Diplomsko delo, Maribor: Univerza v Mariboru, Fakulteta za energetiko, 2012.

[10] (2008) P. L. Joskow, *Lessons Learned From Electricity Market Liberalization.* Dostopno na: http://economics.mit.edu/files/2093 [Dostop 29. 8. 2015]

[11] (2015) Borzen, d.o.o., *Spletna stran družbe*. Dostopno na: https://www.borzen.si/sl/Domov/menu1/Dru%C5%BEba-Borzen/O-dru%C5%BEbi-Borzen [Dostop 13. 9. 2015]

[12] (2009), ERGEG, *Revised ERGEG Guidelines of Good Practice for Electricity Balancing Markets Integration*. Dostopno na: http://www.energy-regulators.eu/portal/page/portal/EER_HOME/EER_CONSULT/CLOSED%20PUBLIC%20CONSULTATIONS/ELECTRICITY/New%20GGP%20Balancing%20Markets%20Integration/CD/E09-ENM-14-04_RevGGP-EBMI_2009-09-09.pdf [Dostop 13. 10. 2015]

[13] (2015) ELES, *Pravila o načinu in pogojih dodeljevanja čezmejnih prenosnih zmogljivosti*. Dostopno na: http://www.eles.si/files/eles/userfiles/avkcije/arhiv-dokumentov/2015%2001%2022%20Pravilnik%20o%20na%C4%8Dinu%20in%20pogojih%2

0dodeljevanja_2015_clean.pdf [Dostop 13. 10. 2015]

[14] (2011) Uradni list RS, št. 97/11, *Pravila za delovanje organiziranega trga z električno energijo*. Dostopno na: http://www.pisrs.si/Pis.web/pregledPredpisa?id=DRUG3257# [Dostop 14. 10. 2015]

[15] V. Rainardi, *Building a Data Warehouse*, New York: Apress, 2008.

[16] W. H. Inmon, *Building the Data Warehouse (3rd Ed.)*, New York: John Willey & Sons, Inc., 2002.

[17] J. O'Brein in G. Marakas, *Introduction to Information Systems*, Columbus: McGraw-Hill Irwin, 2009.

[18] (2015) Investopedia, *Forward Contract*. Dostopno na: http://www.investopedia.com/terms/f/forwardcontract.asp [Dostop 25. 9. 2015]

[19] (2011) D. Stein, *Date vs Integer Datatypes as Primary Key for Data Dimensions.* Dostopno na: http://www.made2mentor.com/2011/05/date-vs-integer-datatypes-as-primary-key-for-date-dimensions/ [Dostop 25. 11. 2015]

[20] C. J. Date, *An Introduction to Database Systems, Eighth Edition*, New York: Pearson Education, Inc., 2003.

[21] (2015) Microsoft, *Columnstore Indexes Query Performance*. Dostopno na: https://msdn.microsoft.com/en-us/library/dn935005.aspx [Dostop 31. 1. 2016]